Extensions de la suite logicielle FORMID pour assister la conception et le suivi de situations pédagogiques exploitant des simulations ou des objets tangibles

Emmanuel Létondor

le cnam
École d'ingénieurs eicnam

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

CENTRE RÉGIONAL ASSOCIÉ DE RHÔNE-ALPES

MÉMOIRE

présenté en vue d'obtenir

le DIPLÔME d'INGÉNIEUR CNAM

SPÉCIALITÉ : INFORMATIQUE

OPTION : Systèmes d'information (ISI)

par

LÉTONDOR Emmanuel

« Extensions de la suite logicielle FORMID pour assister la conception et le suivi de situations pédagogiques exploitant des simulations ou des objets tangibles. »

Soutenu le 25 / 01 / 2016

JURY

PRESIDENT :	**M. Éric GRESSIER-SOUDAN**	**CNAM Paris**
MEMBRES :	**Mme Viviane GUÉRAUD**	**équipe MeTAH du LIG**
	Mme Anne LEJEUNE	**équipe MeTAH du LIG**
	M. Marc FAYOLLE	**CNAM Grenoble**
	M. Philippe GARRAUD	**CNAM Grenoble**

Je dédie ce mémoire

À mon père pour ses encouragements.

À ma femme pour son soutien.

À ma maman, mon frère et ma sœur.

À mes enfants.

REMERCIEMENTS

Je tiens à remercier chaleureusement mes responsables **Mme Anne LEJEUNE** et **Mme Viviane GUÉRAUD** pour leur accueil, leur présence et leur soutien pendant toute la durée de mon stage. J'ai particulièrement apprécié les échanges constructifs et l'intérêt porté à mon travail.

Je tiens à remercier **les membres de l'équipe MeTAH** du LIG pour leur accueil et leur amabilité. Le travail s'est grâce à eux déroulé dans des conditions des plus agréables.

Je remercie tous **les membres du jury** de prendre le temps de me lire et d'assister à ma soutenance. Ce moment est une étape importante dans ma vie professionnelle et l'aboutissement d'un long investissement. Merci à **M. Marc FAYOLLE** d'avoir pris le temps de se déplacer pour venir faire un point sur mon stage, et pour ses précieux avis concernant le mémoire.

Je remercie également mon collègue de bureau pendant 7 mois, **M. Arnaud LE MEILLOUR** qui suit le même cursus CNAM que moi. Il m'a apporté beaucoup par son analyse critique au travers de nos différentes discussions.

Je remercie mon oncle, **M. Dominique LÉTONDOR**, de m'avoir fait profiter de son expertise de la langue française.

Je tiens à signifier toute ma reconnaissance et mon affection à mon père, **M. Yves LÉTONDOR**, qui m'a beaucoup soutenu dans les différentes épreuves jalonnant mon cursus.

Je remercie enfin et surtout ma femme, **Christelle**, pour son soutien sans faille dans cette grande entreprise de l'obtention du diplôme d'ingénieur qui a été initiée en 2008.

LISTE DES ABRÉVIATIONS ET DES ACRONYMES

CSV
Comma-separated values 9
DOM
Document Object Model 15
EIAH
Environnements Informatiques pour l'Apprentissage Humain 4
FIFO
First In First Out 12
FTP
File Transfer Protocol 62
HTTP
Hypertext Transfer Protocol 42
IDE
Integrated Development Environment XI
IP
Internet Protocol 47
JAR
Java Archive 41
Java EE
Java Enterprise Edition 5
JDK
Java Development Kit 18
JPA
Java Persistence API 14
JSON
JavaScript Object Notation 22
JSP
Java Server Page 28
LIG
Laboratoire d'Informatique de Grenoble 71
MeTAH
Modèles et Technologies pour l'Apprentissage Humain 4
MOA
Maîtrise d'Ouvrage 71, 76
MOE
Maîtrise d'Oeuvre 71, 76
MVVM
Model-View-ViewModel 57
ORM
Object-Relational Mapping 14
PC
Personal Computer 85
SQL
Structured Query Language 34
TAD
Théorie Anthropologique du Didactique 31
TCP/IP
Transmission Control Protocol on IP 62
UML
Unified Modeling Language 48
URL
Uniform Resource Locator 38
UT
UnderTracks 9
VLAN
Virtual Local Area Network 45
VM
Virtual Machine 85
WAR
Web ARchive 62
XML
Extensible Markup Language 39
XSL
eXtensible Stylesheet Language 62

GLOSSAIRE

ANNOTATION DU CODE

Le code s'écrit par une liste d'instructions qui doivent être *in fine* exécutées par le microprocesseur. Il est possible d'annoter, c'est-à-dire de marquer une portion de code d'une sorte de commentaire spécial qui pourra être lu par un autre bout de code. Beaucoup de cadriciels permettent l'utilisation des annotations de code pour faciliter le travail du développement, comme « @GET » qui permet de spécifier qu'une fonction doit répondre aux requêtes web standard.

BOÎTE BLANCHE

Un fonctionnement de type boîte blanche signifie qu'il est possible en tant qu'utilisateur d'observer les entrées d'un système, ses sorties, mais également l'intégralité de son fonctionnement interne. Pour permettre cela, le code doit soit être instrumentalisé pour offrir une vision complète de l'état interne depuis l'extérieur, soit être rendu public, comme c'est le cas avec les logiciels Open Source.

BOÎTE NOIRE

Contrairement à l'usage courant qui signifie qu'une boîte noire enregistre tout, en informatique il s'agit de l'observation d'un système dont on peut connaître les entrées et les sorties, mais pas le mode de fonctionnement interne. Un logiciel commercial, dont le code source est habituellement inaccessible pour des raisons évidentes, fonctionne dans ce mode.

CADRICIEL

Surtout appelé *Framework* dans le jargon informatique, un cadriciel est un composant ou un lot de composants logiciels qui aident à concevoir les fondations ou aident à la structuration du logiciel.

CNAM

Le Conservatoire national des arts et métiers, fondé en 1794, est un organisme de formation destiné à permettre aux personnes en activité d'obtenir un diplôme à partir de BAC+2 qui soit reconnu par l'état.

COMET

Comet est un modèle d'architecture logicielle permettant de simuler un canal bidirectionnel entre un client et un serveur web, pour que le serveur puisse prendre l'initiative de contacter le client. Une des implémentations de Comet est le long-polling, décrit ci-après.

HTTP

Le protocole HTTP est utilisé par tous les navigateurs internet pour obtenir les informations des sites web, c'est-à-dire au minimum la page web d'accueil. Il implémente plusieurs méthodes, mais la plus utilisée est « GET » pour obtenir un contenu.

LONG-POLLING

Par cette méthode, le navigateur a en permanence une requête en attente de réponse de la part du serveur. Au bout du temps limite (habituellement 30 secondes), si le serveur n'a rien répondu, la connexion se ferme et une nouvelle requête est lancée. De cette manière le serveur a toujours un canal disponible pour notifier le client.

OPEN SOURCE

Beaucoup de notions existent dans le monde « du libre », mais le principe de base de l'Open Source est que le code source de l'application est public. C'est-à-dire que d'une part chacun peut vérifier que le programme fait bien seulement ce pour quoi il est conçu, et d'autre part chacun peut télécharger les sources, adapter le logiciel et le compiler. Cela est en opposition avec les codes fermés qui fonctionnent comme des boîtes noires, comme cela est la cas avec la plupart des produits commerciaux.

ORM

Le lien objet-relationnel permet d'utiliser une base de données relationnelle comme s'il s'agissait d'une base de données orientée objet. Pour cela, chaque classe a une correspondance dans la base de données, puis un cadriciel permet de manipuler ces objets.

SERVICE WEB

A l'opposé d'un site web qui retourne des pages HTML destinées à être interprétées et affichées pour un humain, un service web est à destination des programmes. C'est-à-dire que toutes les informations sont structurées de manière formelle. Un service web utilise le protocole HTTP, et implémente en général uniquement la méthode « GET ».

SERVICE REST

Un service REST s'apparente à un service web dans le sens où il est destiné à un programme. Cette architecture part du postulat que « tout est ressource », et que chaque ressource a un identifiant unique. Un service REST doit implémenter les quatre méthodes HTTP : « GET » pour lire, « PUT » pour modifier, « POST » pour créer, « DELETE » pour effacer.

VLAN

A l'inverse d'un réseau physique dans lequel il est possible de savoir quelle machine est connectée à quelle autre en suivant les câbles, un réseau virtuel (Virtual Local Area Network) est une manière de simuler un réseau qui n'existerait qu'entre certaines machines choisies, en ignorant le reste d'internet. Cette technologie est très utilisée par les entreprises, pour sécuriser la connexion intersites, ou permettre aux itinérants d'accéder aux ressources de l'entreprise. Le VLAN doit alors être configuré sur leur ordinateur portable ainsi qu'au sein de l'entreprise pour donner l'impression à l'utilisateur que sa machine est présente physiquement dans les locaux. Il a alors accès aux serveurs, aux imprimantes, aux autres machines.

WEBSOCKET

C'est un protocole réseau standard sur le web pour établir un canal de communication bidirectionnel entre un client et un serveur, par-dessus une connexion TCP/IP. Concrètement, de la même manière qu'un navigateur peut contacter un serveur, en utilisant un tel canal un serveur peut contacter un client.

WI-FI

Le Wi-Fi est un ensemble de protocoles de communication sans fil. Il permet de relier plusieurs périphériques pour établir un réseau informatique.

SOMMAIRE

INTRODUCTION

Dans le cadre de l'obtention du titre d'ingénieur Cnam en informatique, il m'est demandé de prouver mes capacités à me comporter en ingénieur. Le contexte de cette réalisation se situe dans une équipe de recherche du laboratoire d'informatique de Grenoble (LIG), spécialisée dans l'analyse et l'amélioration de l'apprentissage humain (MeTAH). Ma mission est de faire évoluer la suite logicielle FORMID (Formation Interactive à Distance). FORMID est une plateforme permettant de travailler sur des situations pédagogiques particulières par le biais de simulations ou de micromondes.

Pour permettre à des élèves d'acquérir un savoir donné, ils sont placés dans des situations d'apprentissage, et des tâches leur sont confiées. Ils utilisent une simulation ou un micromonde, qui leur permet de manipuler des objets comme des billes ou des composants électriques dans le respect des contraintes de l'environnement simulé. Par exemple des composants électriques peuvent griller dans un circuit électrique malformé. L'objectif de FORMID est de contrôler et de piloter la réalisation d'exercices dans ces conditions. Pour ce faire, l'auteur conçoit des scénarios pédagogiques qui s'appuient sur la simulation ou le micromonde concerné. Lors de leur mise en œuvre en classe, la progression des élèves dans les différents exercices est suivie par le tuteur, qui visualise de manière globale les situations observables et les validations demandées. Enfin, toutes les actions qui ont été effectuées par l'auteur et le tuteur sont tracées en vue d'être analysées pour faire progresser la recherche. En effet, la finalité des expérimentations consiste aussi bien à faire évoluer l'apprentissage de l'élève qu'améliorer la facilité d'usage de FORMID.

Je dois dans ces circonstances mener à bien plusieurs objectifs. D'une part il est souhaité que FORMID s'ouvre au monde réel, tangible, en offrant à l'élève la possibilité de manipuler de vraies bûchettes pour apprendre la numération. D'autre part, l'étude de l'apprentissage d'une discipline mène à l'élaboration de modèles praxéologiques. C'est-à-dire la formalisation de données didactiques, contenant notamment les techniques de résolution d'une tâche et les technologies qui les justifient. L'auteur doit avoir accès à ces données pour concevoir plus aisément des scénarios pédagogiques pertinents. Pour aller plus loin, les données praxéologiques qui ont servi à l'élaboration du scénario doivent être mises en exergue dans l'écran du tuteur pour donner une sémantique nouvelle aux observations. Enfin, d'un point de vue plus général, FORMID doit être amélioré pour combler des manques et ajouter de nouvelles fonctionnalités.

Je vais suivre une chronologie sur les trois premiers chapitres. Dans le premier, j'introduirai mon immersion dans le contexte et poserai ainsi les bases nécessaires à la compréhension de l'ensemble du mémoire. Ensuite, dans le second, je détaillerai mes réalisations techniques et fonctionnelles les plus conséquentes. Le troisième chapitre sera quant à lui consacré à ma description de l'état des lieux de FORMID à la fin de mon intervention. Et enfin, dans un quatrième et dernier chapitre, je prendrai de la hauteur sur cet ensemble, pour aborder à la fois la problématique de la gestion de projet et le bilan.

1 IMMERSION

Ce chapitre expose mon immersion dans FORMID, et pose ainsi les bases permettant d'appréhender le chapitre suivant qui traite de mes réalisations. Je vais commencer par énumérer mes objectifs, puis j'introduirai le contexte de travail, avant de détailler l'existant d'où je suis parti. J'exposerai enfin l'attitude que je compte adopter au su de ces informations.

1.1 OBJECTIFS

Les objectifs sont à plusieurs niveaux, et les mettre en perspective me paraît important.

1.1.1 TRAVAIL D'INGÉNIEUR

La raison d'être du mémoire est de prouver que je suis capable de me comporter en ingénieur. J'aspire ainsi à valider mon cursus Cnam, débuté en 2008, par l'obtention du diplôme d'ingénieur en informatique.

Un ingénieur doit, dans la vision que j'en ai, être capable de surmonter tous les obstacles en inventant des solutions. Mais il doit également être capable de s'extraire de la technique pour considérer la vue d'ensemble. Être capable d'analyser une situation de manière cartésienne, factuelle, et pouvoir ainsi la gérer de manière efficiente. Son rôle est également d'organiser, de coordonner et de transmettre, c'est pourquoi la qualité de sa communication est primordiale.

Ce mémoire s'inscrit parfaitement dans cette démarche, en me permettant d'expliquer le travail que j'ai effectué, pourquoi je l'ai fait, et comment. Le comportement d'ingénieur ne se limite donc pas au temps de réalisation, mais concerne le mémoire dans sa globalité, soutenance comprise.

1.1.2 SUJET DU MÉMOIRE

L'environnement de mon travail d'ingénieur a comme support la suite logicielle FORMID dans le laboratoire MeTAH au sein du LIG. Mon intervention a été déclinée dans le sujet du mémoire en quatre grands objectifs.

1. **Évolution de la suite FORMID pour la prise en compte d'objets tangibles**

Le tangible fait référence à ce qui peut être touché, en opposition aux simulations numériques représentant une partie virtuelle du monde. Le présent projet consiste à permettre aux élèves apprenant la numération de pouvoir manipuler des bûchettes. L'objectif est de leur offrir le support des scénarios d'apprentissage FORMID.

2. **Assistance à la conception de scénario : renforcement et nouvelles évolutions**

L'outil FORMID auteur doit être amélioré pour faciliter la conception de scénarios. Ensuite, il est souhaité renforcer la présence de données didactiques dans ce même but. Ces données sont issues d'un serveur de praxéologies interne à MeTAH. L'auteur, avec la connaissance des différentes notions censées être acquises par l'élève, demandées par l'institution dans le cadre des programmes, et une liste des erreurs communes, peut orienter efficacement son scénario.

3. **Assistance au suivi : conception et développement de nouvelles catégories de fonctionnalités**

En poursuivant dans la logique de l'intégration de données didactiques pour aider l'auteur, il est souhaité que ces informations émergent dans le module tuteur pour fournir la sémantique des situations observées. C'est-à-dire présenter au tuteur le contexte praxéologique qui a mené à chaque aspect du scénario.

4. **Amélioration, exploitation en vraie grandeur et documentation de la suite FORMID**

Cette dernière partie regroupe les anomalies et améliorations nécessaires à l'usage courant de la suite logicielle FORMID. C'est-à-dire développer les attentes fonctionnelles manquantes, mais aussi en faciliter l'utilisation. Les parties originellement identifiées étaient le module auteur et le module d'administration.

1.1.3 OBJECTIFS PERSONNELS

Au-delà de mon objectif principal qui est de mener à bien les missions qui me sont confiées de manière professionnelle et rigoureuse, se situent d'autres intérêts et ambitions personnels.

J'ai déjà par le passé eu plusieurs contacts avec le monde Java, mais pas assez pour qu'il soit pertinent de l'inscrire à mon curriculum vitæ. Je souhaite acquérir suffisamment de compétences pour pouvoir prétendre à un poste dans le domaine. Une grande partie des offres d'emploi qui me correspondent dans le développement logiciel est en effet couverte par le duo Microsoft C# et Java EE.

Et enfin, j'ai l'ambition de profiter du stage pour mettre en œuvre une vraie gestion de projet. Je n'ai en effet jamais eu l'occasion de pratiquer cette discipline et j'ai envie d'étendre mon savoir-faire.

1.2 CONTEXTE

1.2.1 CONTEXTE ORGANISATIONNEL

Le Laboratoire d'Informatique de Grenoble, ou LIG [**LIG15**], est structuré en cinq axes de recherches et regroupe 23 équipes pour un total de plus de 500 personnes. Ce mémoire se déroule au sein de l'équipe de recherche "Modèles et Technologies pour l'Apprentissage Humain", MeTAH [**METAH15**], qui regroupe des informaticiens et didacticiens étudiant les Environnements Informatiques pour l'Apprentissage Humain (EIAH) sous la direction de Pierre Tchounikine.

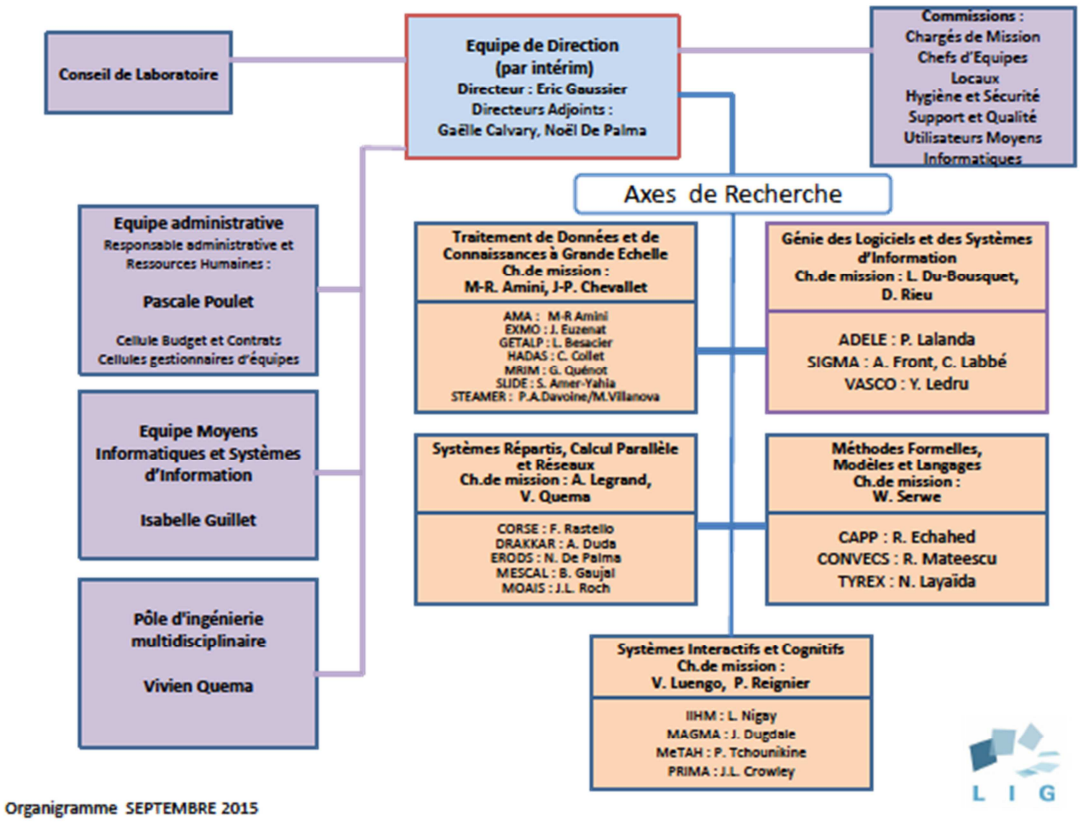


Figure 1 : organigramme du LIG. L'équipe MeTAH est dans le pôle "Systèmes Interactifs et Cognitifs" (en bas).

L'équipe MeTAH est constituée d'une vingtaine de membres permanents et de quinze membres non permanents, dont une majorité de doctorants. Cette équipe travaille, outre sur les projets propres à chacun, à l'élaboration en commun d'un modèle praxéologique, nommé T4TEL, qui doit apporter un socle théorique aux différents EIAH développés. Nous reviendrons sur les praxéologies par la suite.

Au sein de MeTAH, Viviane Guéraud et Anne Lejeune incarnent le groupe le travail qui dirige le développement et l'exploitation de la suite logicielle FORMID, objet de mon travail. Comme cela sera expliqué dans la partie 1.3, l'apprenant y est placé dans une situation d'apprentissage scénarisée, par le truchement d'une simulation ou d'un micromonde.

FORMID a été élaboré en relation avec d'autres groupes de travail, qui ont réalisé notamment le micromonde électrique, utilisé depuis les origines de FORMID, et le serveur de praxéologies. Un travail collaboratif avec Amiqual4home a en outre donné naissance au dispositif tangible.

1.2.2 CONTEXTE TECHNIQUE

L'aspect technique est important car il constitue le support concret du travail. Les technologies utilisées par FORMID ont évolué au fil du temps, pour devenir un projet Java EE côté serveur, qui dispose d'une partie applicative JavaScript côté client. La communication est basée sur des services REST et des WebSockets. L'accès aux données est assuré par une couche de persistance côté serveur. Le système de gestion de base de données utilisé est "H2".

Le serveur de production est une machine virtuelle Linux avec distribution CentOS 6, et il héberge le serveur applicatif Wildfly, requis par FORMID qui utilise la technologie Java EE. Le développement quant à lui se déroule communément dans l'environnement Windows, bien qu'il n'y ait pas de contrainte de ce côté-là.

1.3 FORMID

La suite logicielle FORMID signifie "Formation Interactive à Distance". L'objectif affiché est la capacité de mettre des apprenants dans une situation donnée, et d'observer leur évolution dans le scénario définissant le cadre de travail. Les élèves peuvent travailler pendant une séance dédiée ou de manière asynchrone, et notamment à distance. Cette partie a pour vocation de présenter les grandes lignes de cette suite logicielle FORMID, de manière suffisante à la compréhension de l'ensemble du mémoire. Nous allons donc d'abord étudier la notion de dispositif externe qui sert de support à l'apprentissage, puis celle de scénario pédagogique qui guide l'élève dans son usage du dispositif externe. Forts de ces explications, nous suivrons ensuite le processus de fonctionnement de FORMID. Enfin, je mettrai en avant la finalité du produit.

1.3.1 LES DISPOSITIFS EXTERNES

L'objectif de FORMID est l'utilisation de scénarios pédagogiques dans des situations d'apprentissage. Le théâtre de ces situations est un dispositif externe, c'est-à-dire une simulation ou un micromonde qui a sa vie propre, et auquel FORMID se connecte pour observer les changements d'état.

Le logiciel FORMID a en effet été réfléchi et conçu pour travailler sur des états, et non sur des événements différentiels. C'est-à-dire que ce n'est pas l'action de l'utilisateur qui est surveillée, mais le nouvel état résultant de cette action. Ce choix est primordial, car il détermine le fonctionnement de l'ensemble, depuis la conception de scénario jusqu'à l'étude des traces produites. Cette approche, élaborée à la fin des années 90, n'a pas été remise en cause depuis. [**GUÉRAUD11**]

Le terme de dispositif externe regroupe les notions assez proches de simulation et de micromonde, et a été choisi pour signifier que le dispositif n'est pas développé par FORMID, mais simplement utilisé dans FORMID par le biais d'un mécanisme de connexion. Cette distinction est importante à l'heure où certains dispositifs apparaissent visuellement à l'intérieur de l'application.

1.3.1.1 Les simulations

Les simulations numériques sont des programmes qui traitent des données d'entrée et génèrent des sorties en reproduisant un fonctionnement du monde réel. Elles permettent pour FORMID de manipuler des objets ou des concepts de manière virtuelle, ou, comme nous le verrons, également de manière tangible[1]. Les variables obtenues en sortie sont identifiées à l'avance, seule leur valeur évolue. Un bon exemple de simulation est *TPNum* (voir Figure 2), développé par Timothée Lemaire (stagiaire d'IUT 2 Grenoble, 2015), qui permet de faire travailler la numération aux élèves d'école primaire à l'aide de billes.

[1] Qu'on connaît par le toucher ; matériel, sensible : *la réalité tangible*. (Définition Larousse)

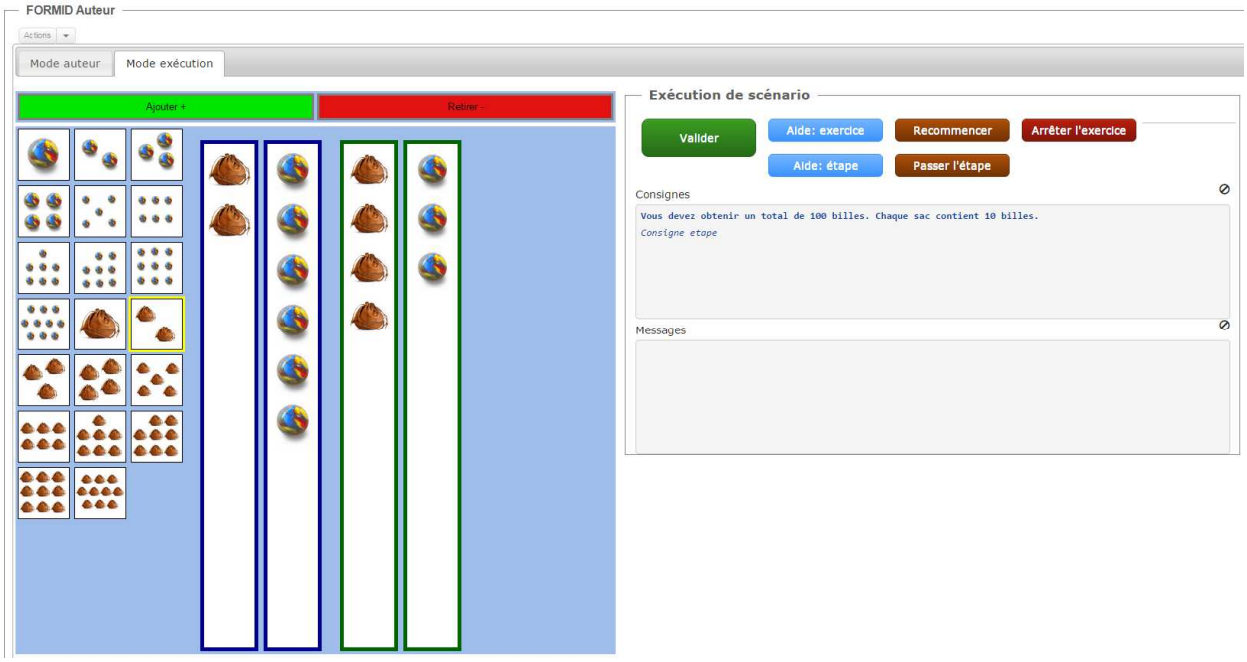


Figure 2: *TPNum* est une simulation permettant d'apprendre la numération. Ici le complément à 100.

Le cadre pour l'élève est posé. L'auteur donne 26 billes comme valeur de départ dans les zones bleues. L'élève peut sélectionner des lots de billes et de sacs sur la gauche, et les ajouter (ou les retrancher) à la partie verte, pour obtenir au total 100 billes.

Variables du dispositif

Nom	Valeur	Valeur précédente	Type
nbBInit	26		int
nbBTotal	69		int
nombreDonne	0		int
nbAction	2		int
defautBilles	faux		String
selectedB	0		int
selectedS	**2**	0	int

Figure 3 : liste des variables de la simulation TPNum.
"nbBInit" est le nombre de billes choisi par l'enseignant, "nbBTotal" le nombre total actuel, "nbAction" le nombre d'ajouts et de soustractions effectués, "selectedB" le nombre de billes sélectionnées, "selectedS" le nombre de sacs sélectionnés. Ce sont les dernières variables modifiées qui apparaissent en gras.

La liste des variables, représentée Figure 3, est invariante pendant l'utilisation par l'élève.

1.3.1.2 Les micromondes

Les micromondes permettent de mettre en œuvre des méthodes de pédagogie active amenant l'apprenant à créer et à modifier des éléments à l'intérieur de celui-ci. D'un point de vue plus pragmatique, ce qui différencie une simulation d'un micromonde, c'est le fait que le nombre de variables observables dans un micromonde n'est pas prévisible. En effet, chaque nouvel élément ajouté dans le micromonde se verra caractérisé par une ou plusieurs variables. Si l'on imagine un village, l'ajout d'une maison donnera naissance à des variables pour l'adresse, le type, la couleur, le nom du propriétaire.

Un bon exemple de micromonde est *TPElec* [**TPELEC15**] illustré Figure 4, qui représente une table sur laquelle il est offert la possibilité de construire un circuit électrique.

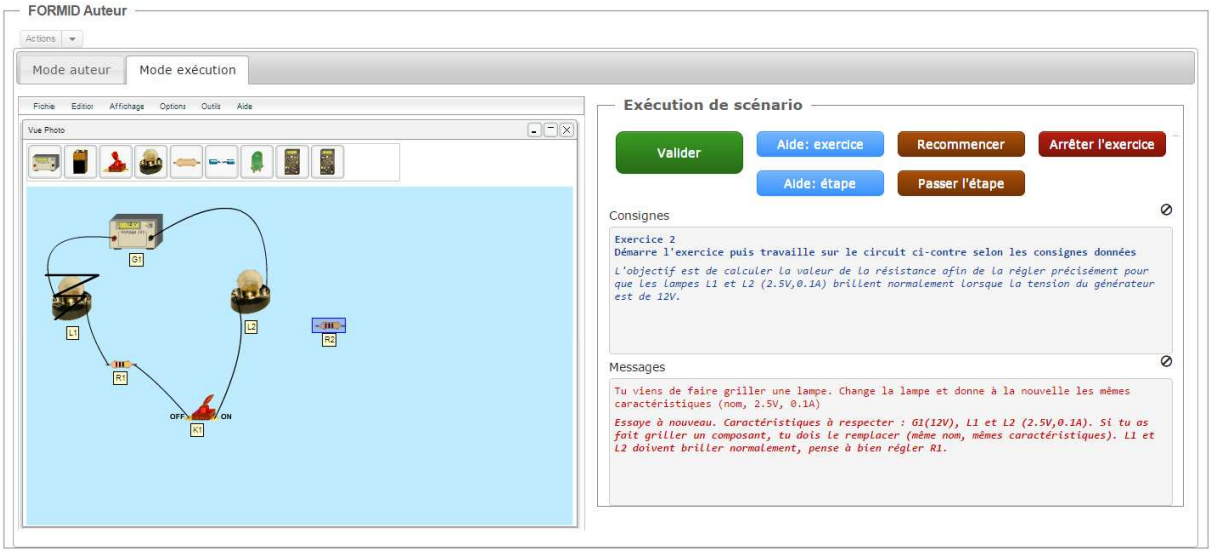


Figure 4 : le micromonde TPElec permet de créer un circuit électrique

L'apprenant peut ajouter des éléments et les relier par des fils, comme le montre l'ajout de la résistance R2. La Figure 5 montre que cet ajout a pour conséquence l'ajout de cinq nouvelles variables liées à R2.

Variables du dispositif

K1_CLAQUE	non		String
K1_COURANT	0		Double
K1_TENSION	0		Double
K1_ETATI	oui		String
K1_RESISTANCE	0.1		Double
L1_CLAQUE	oui		String
L1_COURANT	0		Double
L1_TENSION	0		Double
L1_INTN	0.1		Double
L1_PUIN	0.25		Double
L1_TENN	2.5		Double
L2_CLAQUE	non		String
L2_COURANT	0		Double
L2_TENSION	0		Double
L2_INTN	0.1		Double
L2_PUIN	0.25		Double
L2_TENN	2.5		Double
R2_CLAQUE	**non**		String
R2_COURANT	**0**		Double
R2_TENSION	**0**		Double
R2_PUIN	**1**		Double
R2_RESISTANCE	**100**		Double

Figure 5 : liste des variables du micromonde TPElec affiché Figure 4.
Les variables disponibles dans le micromonde TPElec dépendent du circuit de l'apprenant. Ici la résistance R2 vient d'être ajoutée. "R2_CLAQUE" indique si la résistance a grillé, "R2_COURANT" est l'intensité en ampères, "R2_TENSION" la tension en volts, "R2_PUIN" la puissance maximale dissipée en watts, et "R2_RESISTANCE" la valeur de la résistance en ohms.

Il est donc naturellement plus difficile de cadrer les scénarios d'apprentissage dans un micromonde que dans une simulation, puisque les variables en présence ne sont pas connues a priori.

1.3.2 LE SCÉNARIO PÉDAGOGIQUE

Le scénario pédagogique est au centre de FORMID. Il représente un exercice composé d'étapes que l'apprenant doit réaliser avec un dispositif externe. Concrètement, nous pouvons placer l'élève dans un contexte d'électricité. Nous pouvons alors lui demander de calculer la résistance permettant de protéger une ampoule, puis par la suite d'ajouter un voltmètre pour contrôler la tension à ses bornes (Cf. Figure 6 gauche).

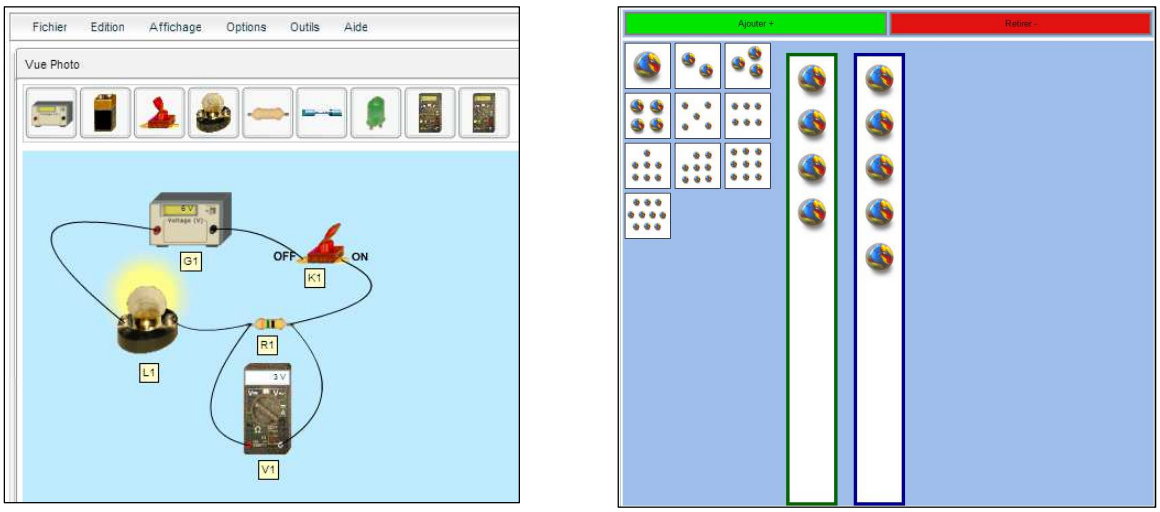


Figure 6 : dispositif externe de type "électricité" *TPElec* (à gauche), et de type "billes" *TPNum* (à droite).

Les variables du système sont observées pendant que l'élève manipule, et des contrôles préétablis peuvent se déclencher. Ce sont des situations observables déterminées par l'auteur du scénario. Les validations d'étapes quant à elles doivent être demandées par l'apprenant. Une évaluation de contrôle ou d'étape consiste à tester une combinaison de variables, par exemple tester s'il y a plus de 8 billes au total (Cf. Figure 6 droite).

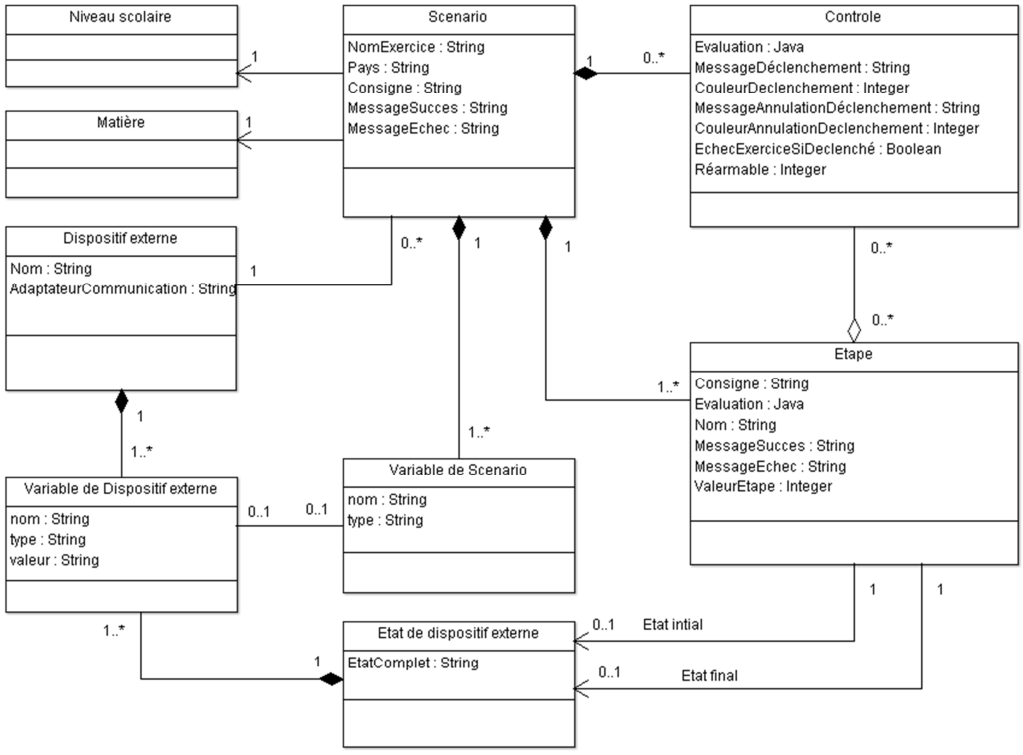


Figure 7 : représentation UML d'un scénario pédagogique FORMID

Le diagramme de classes Figure 7 décrit la composition d'un scénario pédagogique de manière plus formelle. L'application FORMID ne se limite pas, en général, à un rôle d'observateur du dispositif externe. C'est pourquoi le scénario contient l'état du dispositif externe à affecter pour chaque étape qui le nécessite.

Nous allons maintenant voir comment les modules de FORMID s'articulent autour de ce scénario pédagogique.

1.3.3 PROCESSUS DE FONCTIONNEMENT

La suite logicielle FORMID est composée de quatre modules. Le premier, d'administration, est un passage obligé dans les logiciels de gestion. Il permet la saisie de toutes les données de base utilisées dans les autres modules. Ensuite le module auteur permet la conception de scénarios pédagogiques, le module élève permet l'utilisation de ce scénario par un apprenant, et enfin le module tuteur sert à suivre un groupe réalisant un ensemble d'exercices. Nous devons finalement évoquer l'export des traces produites depuis le module d'administration.

1.3.3.1 Administration

La partie d'administration de FORMID permet de saisir les données basiques. Il s'agit de paramétrages incontournables vitaux au bon fonctionnement des autres modules.

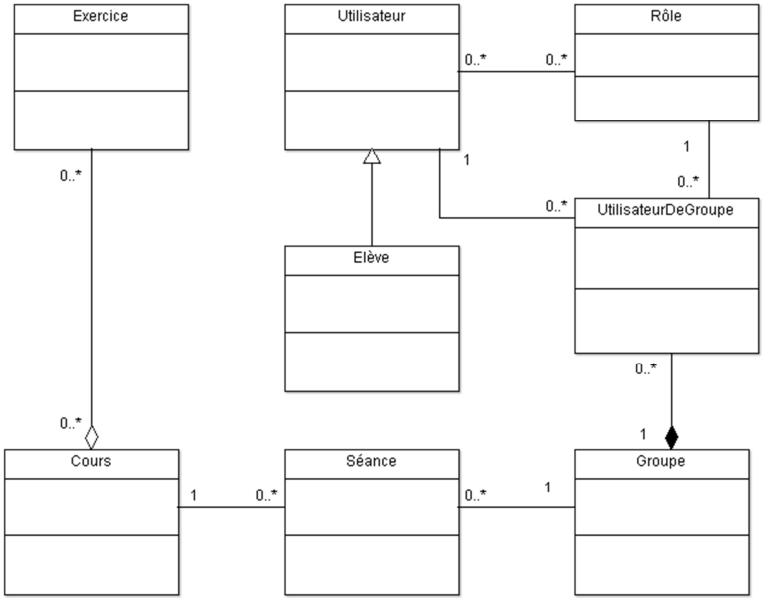


Figure 8 : diagramme UML des données gérées par le module d'administration.

La Figure 8 montre la partie du modèle de données gérée ce module. Tout d'abord, un utilisateur possède un ou plusieurs rôles, dont principalement administrateur, auteur, élève, tuteur. Si l'utilisateur est un élève, il est possible de renseigner des données supplémentaires : notamment sa photographie, voire un pseudonyme permettant l'anonymisation des données. Ensuite les modules élève et tuteur travaillent à partir d'une séance, c'est-à-dire le fait de dispenser un cours, constitué d'exercices, pour un groupe d'élèves à une date donnée. En synthèse, la saisie se compose de la gestion des cours d'un côté et de la gestion des groupes de l'autre, qui se rejoignent pour les séances. Le module d'administration permet d'effectuer des opérations d'export des traces élève et tuteur dans un fichier texte au format CSV. Toutes les actions jugées pertinentes sont en effet enregistrées dans des tables dédiées. L'export de ces données permet leur visualisation dans un tableur ou leur import dans UnderTracks (UT), un outil MeTAH dédié à l'analyse des traces.

1.3.3.2 Module auteur

Avec le module auteur nous entrons dans le vif du sujet, puisque nous mettons en lien les notions de dispositif externe et de scénario pédagogique. Le concept de base est le suivant. FORMID est connecté, par un biais technique quelconque, à un dispositif externe, qui peut donc informer de chaque changement de son état. L'auteur agit sur le dispositif externe directement, et il observe les variables changer de valeur. Il détermine les variables pertinentes pour son scénario et crée les étapes de résolution de l'exercice.

Le scénario pédagogique modélisé Figure 7 (page 8) se présente à l'auteur sous la forme d'un arbre. À côté se présente le dispositif externe choisi. La Figure 9 montre un exemple de scénario pédagogique portant sur le micromonde électrique *TPElec*.

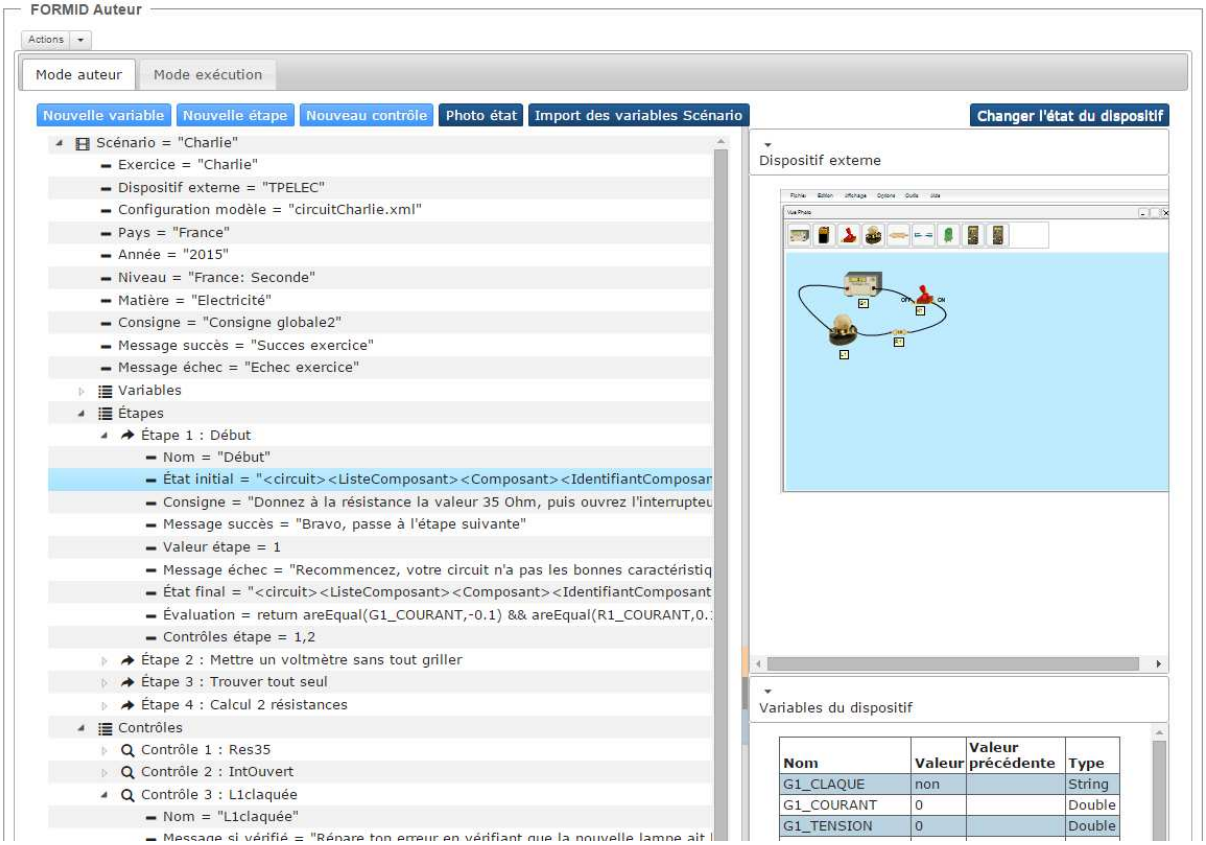


Figure 9 : le module Auteur.
L'arbre du scénario (à gauche) est entièrement modifiable, il porte ici sur le dispositif externe "*TPElec*" (à droite).
Les variables obtenues (extrait en bas à droite) sont mises à jour lors de chaque manipulation dans le dispositif externe.

L'auteur crée chaque étape en saisissant une consigne pour l'élève et une condition d'évaluation du succès de l'étape en langage Java. Il peut également créer des contrôles évaluant des situations attendues pour guider l'élève.

L'auteur dispose d'un onglet "Mode exécution" qui lui permet de tester immédiatement son scénario. Cet aller-retour entre le mode conception et le mode exécution est très facile et permet des mises au point rapides.

Une fois le scénario finalisé par l'auteur, il peut le publier dans la base de données. Pour être utilisable, l'exercice doit ensuite être inscrit à un cours depuis le module d'administration. Une séance pour ce cours doit également être planifiée. Nous allons maintenant voir comment cet exercice créé par l'auteur est réalisé par l'élève.

1.3.3.3 Module élève

Un élève se connecte à FORMID avec ses identifiants. Il choisit son groupe, la séance de travail, et enfin l'exercice qu'il souhaite réaliser. L'application est donc prête à démarrer. L'élève clique sur le bouton "Démarrer" et le dispositif externe s'affiche dans l'état initial, alors que les premières consignes d'exercice et d'étape s'affichent.

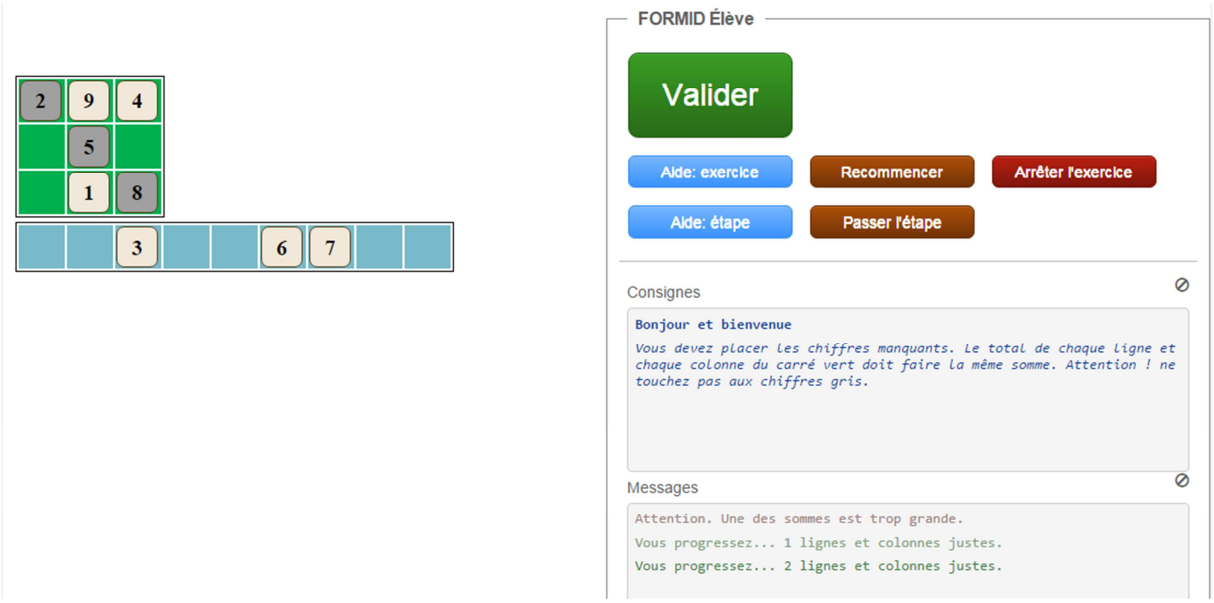


Figure 10 : interface du module élève sur le dispositif externe "Jeu de lettres" (*LettersGame*).
L'élève clique sur "Valider" à chaque fois qu'il pense avoir répondu correctement aux consignes de l'étape. Il peut également choisir de recommencer l'étape ou de la passer. Dans ce scénario, un contrôle affiche un message à l'élève lorsqu'il avance dans l'exercice.

À partir de ce moment-là, l'élève a démarré le scénario pédagogique dans la mise en situation du dispositif externe (Figure 10). Le déclenchement des contrôles et les actions initiées par l'élève, comme la validation d'étape, sont consignés pour le module tuteur et pour l'export, avec toutes leurs caractéristiques et l'état correspondant du dispositif externe.

Nous allons maintenant voir le module tuteur, qui permet de piloter un groupe de travail qui réalise des exercices.

1.3.3.4 Module tuteur

Un groupe de travail se compose de nombreux élèves, qu'il est important de pouvoir suivre dans une vue d'ensemble, pour constater l'avancement général. Cela permet d'aider les élèves en difficulté, et de comprendre quelles erreurs sont le plus fréquemment commises.

Le module tuteur offre trois vues de granularités différentes. La vue globale, présentée Figure 11, donne une vision claire de la progression de tous les élèves du groupe. Des infobulles[2] sont disponibles sur tous les carrés et rectangles de couleur, et affichent les valeurs des variables choisies par l'auteur au moment de la conception du scénario. Les deuxième et troisième vues (Cf. Annexe 1) entrent dans le détail des contrôles déclenchés. Alors que la deuxième vue affiche tous les événements de tous les élèves sans relation temporelle entre différents contrôles, la troisième met en avant la chronologie des déclenchements pour un élève choisi.

[2] Message d'aide contextuel apparaissant lorsque le curseur est pointé sur certains éléments de l'interface graphique d'un logiciel. (définition Larousse)

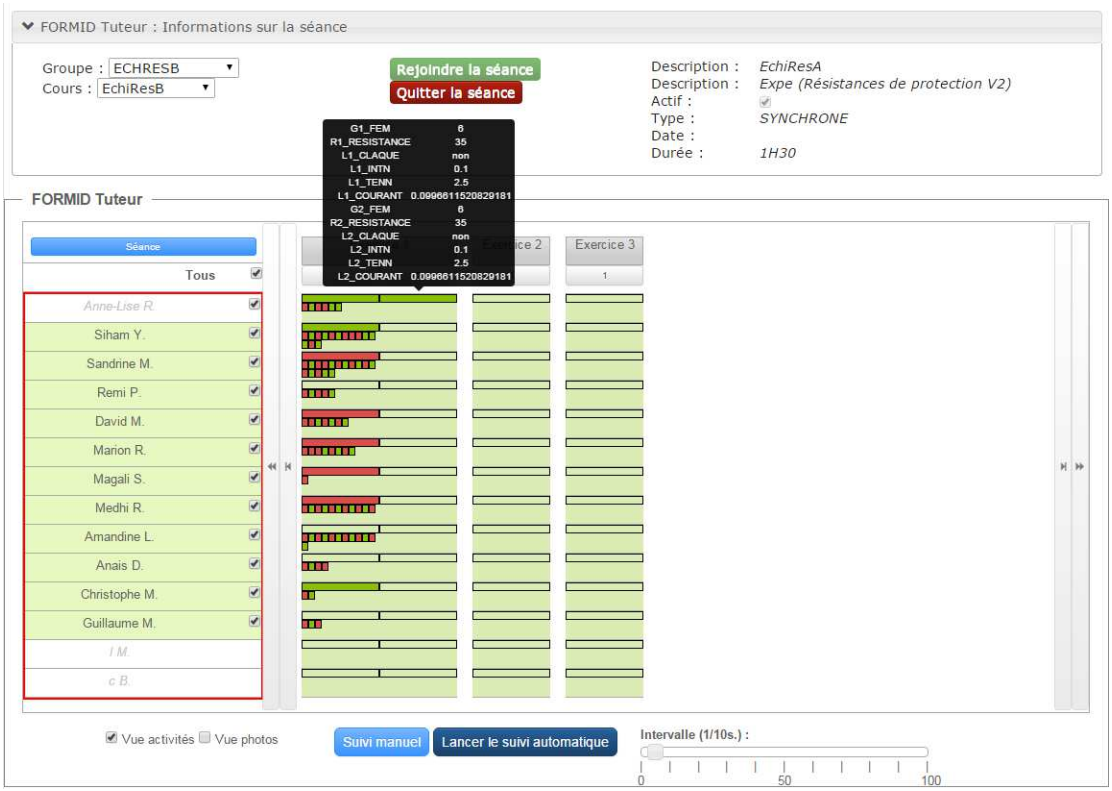


Figure 11 : vue globale du module tuteur.
Tous les élèves du groupe sont représentés.
Les grands rectangles indiquent la dernière tentative de validation de l'étape (rouge échec, vert succès).
Les petits carrés représentent les contrôles déclenchés (rouge erreur commise, vert information ou encouragement).

D'un côté, chaque fait notable de chaque élève sur une séance est écrit en base de données de manière séquentielle. Et de l'autre, lorsque le tuteur visualise une séance en mode de suivi automatique, une information est lue à intervalles réguliers. Les deux mécanismes peuvent être conceptuellement considérés comme une liste FIFO (le premier élément inscrit sera lu en premier). Le tuteur peut donc voir tous les événements dans l'ordre, à la vitesse qui lui convient. Et s'il est connecté pendant que les élèves travaillent, il les verra en direct. Ce mécanisme est donc asynchrone, aussi bien adapté à un mode d'apprentissage à distance qu'en présentiel.

1.3.4 ANALYSE DES ACTIONS

Les traces ont été mentionnées précédemment dans l'explication des modules élève et tuteur. Il est important de bien considérer que les écritures des événements ne sont pas anecdotiques. L'objectif est en fait double.

Tout d'abord, le tuteur utilise les informations à sa disposition pour suivre les élèves. Par son analyse visuelle, il peut par exemple constater qu'un élève valide systématiquement plusieurs fois avant de trouver la solution, ou que de nombreux élèves déclenchent le même contrôle, ce qui pourrait indiquer qu'ils butent sur les mêmes écueils.

D'un autre côté, de manière différée, une analyse des traces revêt une réelle valeur pour la recherche. C'est pourquoi les élèves ne sont pas les seuls à être scrutés. Les tuteurs et les auteurs aussi génèrent des traces, pour pouvoir optimiser l'utilisabilité générale du module concerné. Il est possible de récupérer les traces produites dans un tableur, ou dans la plateforme d'analyse de traces UnderTracks [**UNDERTRACKS15**]. Cet outil permet de créer des tables d'événements horodatés, avec des champs paramétrables par application. Une fois ces tables peuplées, de multiples analyses sont possibles.

1.4 HÉRITAGE

1.4.1 PROBLÉMATIQUES DE MES PRÉDÉCESSEURS

S'intéresser à l'historique d'un produit paraît incontournable pour comprendre pourquoi il est dans l'état observé. Cela permet une critique constructive et la poursuite éventuelle de la progression qui a été initiée.

Le logiciel FORMID, piloté par Anne Lejeune et Viviane Guéraud, est essentiellement développé par des stagiaires de formation et d'expérience hétérogènes, et pendant des durées très variables. Sans entrer dans un inventaire des intervenants, il me semble incontournable de citer Patrick Echterbille [**ECHTERBILLE14**]. En effet, il a effectué des modifications structurelles majeures dans FORMID. Ses choix techniques et organisationnels font partie de l'héritage que j'ai reçu.

1.4.1.1 Élimination du composant Java

Tout d'abord, l'algorithme principal gérant les scénarios pédagogiques était implémenté dans un composant Java présent dans le navigateur. Le composant a finalement été totalement éliminé et le moteur de l'algorithme déplacé côté serveur, ce qui a induit plusieurs évolutions : une communication par WebSocket [3], le choix du serveur Wildfly et de la base de données H2 [**H2DATABASE15**]. C'est également ainsi qu'a été introduit le cadriciel AngularJS (expliqué dans la partie 3.1.1) pour structurer l'application du côté client. Au cours de ce mémoire, j'expliquerai que j'ai dû éliminer certaines de ces nouveautés pour faire face à d'autres problématiques.

1.4.1.2 Contrôle de code source Git

Mon prédécesseur a voulu centraliser et sécuriser les sources par la mise en place du gestionnaire de code source Git [**GIT15**] dont le dépôt central est dans la forge Imag [**FUSIONFORGE15**]. Le système Git, créé par Linus Torvalds (créateur du noyau Linux), est décentralisé, et protège aussi bien de la perte du serveur que de celle du poste de développement.

1.4.1.3 Assistance de l'auteur fondée sur des praxéologies

Les praxéologies seront détaillées par la suite ; il s'agit d'un modèle représentant les techniques qui doivent être réalisées pour résoudre un certain type de problèmes, les technologies mises en œuvre, et les erreurs fréquentes. L'auteur peut se baser sur cette connaissance pour établir un scénario pertinent qui observe les situations importantes ou à risque. Le développement en était à l'état d'un premier prototype fonctionnel.

1.4.2 CADRICIELS ET COMPOSANTS LOGICIELS

Dès la prise de contact avec le code source de l'application FORMID, il apparaît que cette application utilise de très nombreux composants logiciels. Nous pouvons les séparer en deux groupes, d'un côté les composants côté client, dans le navigateur internet, et de l'autre ceux du côté serveur, en Java EE.

[3] WebSocket est un protocole réseau qui permet de créer un canal de communication web full-duplex.

1.4.2.1 Composants logiciels côté serveur

Les composants côté serveur sont en nombre assez restreint, faciles à identifier et à justifier.

Tableau 1 : composants logiciels côté serveur

Nom du composant	Usage
Hibernate	Respecte la norme JPA et permet la persistance dans une base de données, en simulant la manipulation d'une base de données orientée objet (méthode ORM). [**HIBERNATE ORM15**]
Atmosphere	Améliore la gestion de la communication client-serveur par WebSockets et gère un mode dégradé. Cela offre la possibilité au serveur de prendre l'initiative de joindre le client. [**ATMOSPHERE15**]
Weld - CDI	Implémentation de référence pour l'injection de dépendances en Java. [**WELD15**]
Jersey	Implémentation des services Web "RESTful" en Java qui respecte la spécification JAX-RS. Offre des possibilités d'implémentation simples par annotation du code. [**JERSEY15**]
Spring	Composant Java permettant l'inversion de contrôle (IoC), donc un pilotage par le cadriciel, facilitant la prise en charge de JPA. [**SPRING15**]

Le Tableau 1 énumère en fait les constituants d'une application Java EE : une couche de services web, renforcée par une persistance en base de données. La gestion des WebSockets est facilitée par une librairie dédiée.

1.4.2.2 Composants logiciels côté client

Du côté client, l'identification des composants a été plutôt complexe. En effet, le système de stockage choisi consistait à répartir les différents fichiers d'un composant par type de fichier. Devant la difficulté de repérage et de mise à jour des composants, j'ai choisi de les réorganiser en laissant tous les fichiers d'un même composant dans un dossier unique. La réorganisation a fait apparaître dix-huit composants logiciels, représentés en Tableau 2.

Tableau 2 : liste des composants logiciels côté client.

Nom du composant	Usage
AngularJS [**ANGULARJS15**]	Cadriciel structurant qui permet d'établir un modèle MVC (Modèle Vue Contrôleur) côté client, mais qui gère également principalement la liaison de données et les modèles de vues.
Angular UI Bootstrap [**UI BOOTSTRAP15**]	Ajout de composants BootStrap fonctionnant sous AngularJS. Gère notamment dans FORMID les infobulles en mode tuteur et les fenêtres modales en mode auteur.
Angular UI ng-grid [**ANGULAR UI GRID15**]	Une partie de la suite "Angular UI" qui permet de gérer des grilles de saisie compatibles avec AngularJS. Elles sont utilisées par exemple dans la partie d'administration de FORMID.
Bootstrap [**BOOTSTRAP15**]	Bibliothèque graphique permettant de gérer l'apparence de manière générale. Mise en page, style des boutons. Contient du JavaScript.
Font awesome [**FONT AWESOME15**]	Polices de caractères qui sont en fait des collections d'icônes. Permet d'intégrer facilement des images qui resteront cohérentes. La taille et la couleur sont modifiables comme pour une police standard.
Html5Shiv [**HTML5SHIV15**]	Permet de faire fonctionner les éléments Html5 inconnus dans Internet Explorer, entre des versions 6 à 9.
JavaScript Canvas to Blob [**JAVASCRIPT-CANVAS-TO-BLOB15**]	Permet de sauvegarder des données dans un fichier binaire, depuis le navigateur. Est utilisé pour sauvegarder les traces tuteur dans la partie administration, et pour enregistrer le scénario en cours dans le mode auteur.

Nom du composant	Usage
JavaScript Load Image [**JAVASCRIPT LOAD IMAGE15**]	Permet d'envoyer une image depuis le navigateur vers le serveur, après avoir offert la possibilité de la redimensionner et de n'en choisir qu'une partie. Utilisé pour la photo des élèves.
jQuery [**JQUERY15**]	Incontournable bibliothèque de fonctions de manipulation du DOM sur de multiples navigateurs.
jQuery Atmosphere [**JQUERY ATMOSPHERE15**]	Partie client d'un gestionnaire de communication par WebSockets gérant les modes dégradés comme le "long-polling".
jQuery custom content scroller [**JQUERY CUSTOM CONTENT SCROLLER15**]	Permet une gestion fine de l'apparence et du fonctionnement des barres de défilement. Utilisé dans le module tuteur.
jQuery file upload [**JQUERY FILE UPLOAD15**]	Améliore la méthode d'envoi de fichiers depuis le navigateur vers le serveur.
jQuery UI [**JQUERY UI15**]	"UI" signifie "User Interface", ou interface utilisateur en français. Il s'agit d'une bibliothèque contenant de nombreux composants visuels utilisant sur la librairie jQuery.
jQuery UI Layout [**JQUERY UI LAYOUT PLUG-IN15**]	Gestionnaire de mise en page avec possibilité d'imbrication des zones de contenu.
jsTree [**JSTREE15**]	Affichage et gestion d'un arbre de données hautement personnalisable. Est utilisé pour représenter le scénario dans le module auteur.
Restangular [**RESTANGULAR15**]	Ajoute des méthodes de gestion d'appel à des services REST avec AngularJS.
sessionStorage [**HTML5 SESSIONSTORAGE15**]	Avec Html5 est apparue la possibilité de stocker des données "de session", côté client, dans un emplacement nommé le "sessionStorage". Cette librairie permet ce fonctionnement sur des navigateurs qui ne le supportent pas nativement.
Underscore.js [**UNDERSCORE.JS15**]	Librairie ajoutant une centaine de fonctions utilitaires globales. Il est difficile de localiser l'usage de ces fonctions qui ont des noms communs, comme "each" ou "filter".

Les éléments sont donc de trois types : le cadriciel structurant AngularJS, des librairies de gestion de l'apparence, et des librairies utilitaires.

1.4.2.3 Réflexion

FORMID utilise de très nombreux composants logiciels, et subit donc autant de dépendances. Se posent naturellement plusieurs questions que nous allons passer en revue.

a) Mises à jour

Un logiciel vit et évolue. Il en est de même de tous les composants utilisés par ce logiciel. Afin qu'il ne se retrouve pas dépassé techniquement, ce qui pourrait nécessiter une importante intervention, une stratégie envisagée pourrait être de maintenir ces dépendances à jour. Mais j'ai dû abandonner cette idée, parce que cela demandait beaucoup trop de temps et présentait trop de risques de régression, sans répondre à un besoin particulier. J'évoquerai plus en détail le problème de l'évolution des composants dans la partie 4.1.4.

b) Évolution de l'interface utilisateur

Il est parfois très difficile de savoir comment l'affichage est organisé (le "*layout*" en anglais). En effet, l'empilement des éléments constituant une page web et la diversité des composants utilisés rendent très compliquée la moindre intervention (Cf. Figure 12).

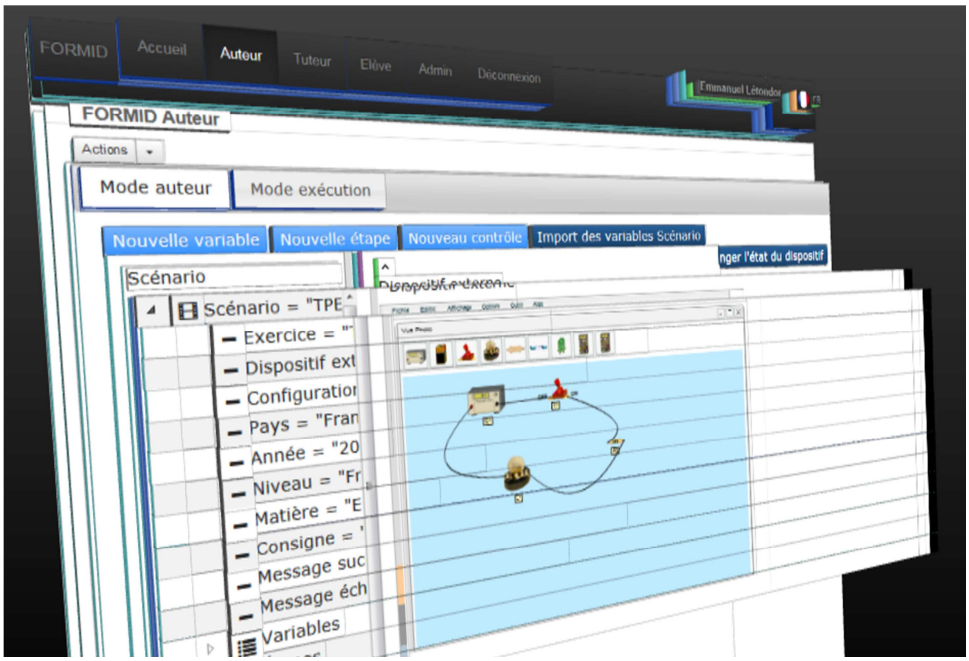


Figure 12 : illustration de l'empilement d'éléments à positionner dans une page web FORMID.
Ici un scénario dans le module auteur (représentation 3D de Mozilla FireFox).

Il n'est pas facile de savoir si la mise en page d'un élément est gérée par le composant « Bootstrap », « AngularJS Layout » ou « jQuery UI ». Dans une même page nous pouvons constater l'utilisation de composants différents en fonction des strates.

c) Barrière à l'entrée

Développer dans un produit créé avec des composants logiciels peut requérir de la part de l'intervenant des connaissances qui vont bien au-delà du langage nu (Java, JavaScript…). Lorsqu'une vingtaine de composants sont listés comme ici, la barrière à l'entrée peut sembler vraiment conséquente. L'apprentissage d'AngularJS est incontournable, les autres composants peuvent potentiellement être appréhendés au besoin.

1.4.3 DÉMARRAGE

C'est la phase pendant laquelle s'effectue la montée en compétences par la maîtrise du code source, de l'environnement logiciel et des outils. Elle s'est concrètement effectuée sur les quatre premières semaines.

1.4.3.1 Mise en route

Je définis le terme de "mise en route" par le fait d'être capable de démarrer et d'arrêter le logiciel par le biais de son code source. Cela implique de savoir manipuler les logiciels requis : donc a minima le serveur applicatif, le système de gestion de base de données, le contrôle de code source.

C'est une phase pendant laquelle il est nécessaire de savoir se familiariser avec les documentations existantes, mais également durant laquelle j'ai pris soin de consigner toutes les informations que j'ai découvertes.

Ce que j'ai considéré comme la mise en route de FORMID m'a demandé 36 heures sur les trois premières semaines. Je vais maintenant en aborder le contenu, puis j'enchaînerai sur la formation qui y est fortement liée.

1.4.3.2 Documentation

La documentation mise à disposition comprend notamment le rapport final de chaque stagiaire, ainsi que la documentation utilisateur complète de référence [**CAGNAT06**]. La partie fonctionnelle étant parfaitement maîtrisée par Anne Lejeune et Viviane Guéraud, je n'avais aucune inquiétude de ce côté-là et aucun problème pour obtenir des explications. C'est pour la découverte de la partie technique que j'avais besoin de beaucoup d'informations. Le Tableau 3 liste les sources encore techniquement valides.

Tableau 3 : liste des documents contenant des informations techniques encore valides

Nom du document	Date	Auteur	Pages	Type
FORMID ajout d'une simulation	27/01/2015	Timothée Lemaire	6	Documentation technique de l'ajout d'une simulation
Développement d'une simulation pour la suite logicielle FORMID	08/01/2015	Timothée Lemaire	29	Rapport de stage
Mémoire d'ingénieur Cnam	03/06/2014	Patrick Echterbille	104	Mémoire d'ingénieur
Dossier d'Architecture Technique	30/04/2014	Patrick Echterbille	27	Documentation technique

Les documents mis à disposition contiennent des informations précieuses, notamment concernant la mise en route de GIT, de H2, de l'architecture REST et des WebSockets, mais ils ne comblent pas tous les besoins. C'est pourquoi pendant toute la durée de mon stage, j'ai consciencieusement consigné toutes les informations qui m'auraient été utiles. Cette démarche et les résultats afférents sont décrits dans la partie 3.4.4.

La mise en route du poste de travail et de l'environnement de développement ne s'est pas terminée avec la mise en application de la documentation. Bien d'autres étapes sont en effet nécessaires, c'est ce que je décris dans la partie suivante, l'installation.

1.4.3.3 Installation

L'installation du poste de travail nécessite nombre de manipulations. Le poste était certes opérationnel, mais la création de mon identité m'a amené à aborder beaucoup de sujets.

a) Environnement de développement NetBeans

Tout d'abord, la nouvelle session Windows m'a contraint à réinstaller les composants additionnels ("*plugins*") de l'environnement de développement NetBeans. Cela est assez coûteux en temps, puisqu'il y avait tout de même 53 composants installés sur la session Windows de mon prédécesseur, et que la comparaison ligne à ligne entre ma version de NetBeans et des copies d'écran est un peu fastidieuse. De plus, tous les composants ne sont pas disponibles dans le dépôt standard, c'est-à-dire que certains doivent être recherchés sur internet.

b) Outil de gestion des dépendances Maven

Ensuite, je me suis heurté à un problème lors de la tentative d'exécution de FORMID. Le message d'erreur indiquait que "certains artéfacts de dépendances ne sont pas dans l'entrepôt local" (message d'origine : "*some dependencies artifacts are not in the local repository*"), ce qui m'était au départ incompréhensible. J'ai donc découvert la nécessité d'installer et de configurer Maven [**APACHE MAVEN15**]. C'est un outil puissant qui va télécharger récursivement les dépendances nécessaires au bon fonctionnement du logiciel, et générer le fichier de déploiement pour la mise en production.

c) Gestionnaire de code source GIT

Le gestionnaire de code source a comme buts de conserver toutes les versions de chaque fichier du programme, et de garder l'historique de qui a fait quoi, et quand. Très utile pour travailler seul, il devient indispensable dans une équipe, pour éviter qu'un développeur n'écrase le travail d'un autre par inadvertance.

Grâce au tutoriel de configuration de Git, cette partie a pu être assez facilement réalisée, bien qu'elle comprenne de nombreuses étapes. Cependant, Git étant un gestionnaire de code source décentralisé, notion complètement nouvelle pour moi, j'ai dû après l'installation me former à son utilisation courante.

d) NetBeans

Un environnement de développement permet de gérer le code source du programme, c'est-à-dire de le taper, de le modifier, et de le compiler pour l'exécution. NetBeans, qui est utilisé pour FORMID, m'a posé des problèmes. Tout d'abord, il m'a fallu comprendre que très régulièrement, lorsque plus rien ne semble fonctionner, il s'agit d'un problème de remplissage de la mémoire, et que la solution est de quitter NetBeans, voire de tuer le processus Java, pour résoudre le problème.

Ensuite, j'avais pendant une période des problèmes au lancement de NetBeans, qui ne démarrait plus. J'ai découvert qu'en vidant le cache interne du logiciel, cela fonctionnait, mais qu'au lancement suivant, le problème réapparaissait. Au final il s'agit d'un problème de JDK (Kit de développement Java). Pour une raison que je n'ai toujours pas trouvée, NetBeans en cherchait une ancienne version dans le disque dur. Mettre à l'emplacement en question la version courante du JDK a résolu le problème. J'ai bien conscience que ce n'est pas une solution propre ou pérenne, mais j'ai passé beaucoup de temps sur les forums sans résultat avant de tester cette méthode palliative.

1.4.3.4 Formation

La formation peut être nécessaire pour combler certaines lacunes et permettre de démarrer dans de bonnes conditions. Cela est un investissement à doser pour obtenir une bonne compréhension du sujet et être capable d'écrire soi-même de nouvelles lignes de code. Il ne faut pas non plus chercher à dominer complètement le sujet, puisque le temps qu'il faudrait y passer serait démesuré. Malgré mes années d'expérience, les sujets n'ont pas manqué. La Figure 13 montre les différentes disciplines et la proportion de temps consacré à chacune.

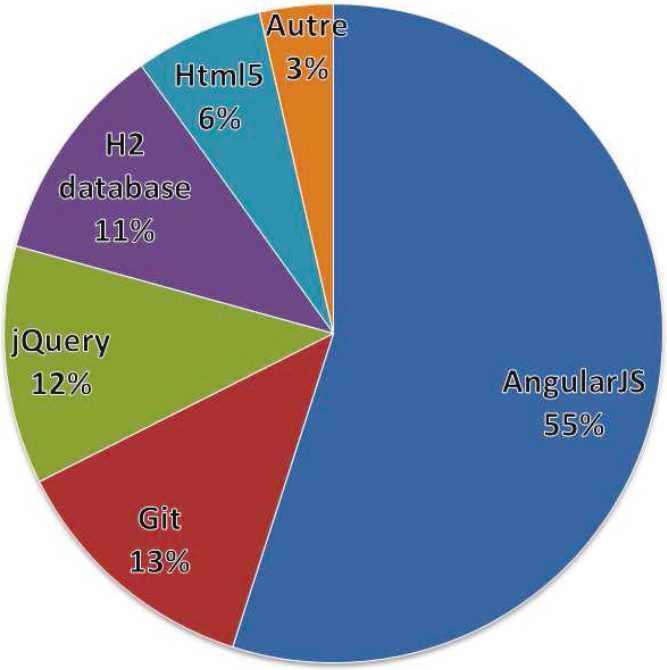


Figure 13: répartition de la durée d'autoformation par technologie. La durée totale est de 55h30.

Le temps d'autoformation que j'ai mesuré ne comprend que les heures qui ont été entièrement dédiées à l'apprentissage et à l'expérimentation d'une discipline. En effet, l'apprentissage induit par la pratique quotidienne est difficilement chiffrable, puisqu'il est confondu avec du développement productif. J'ai finalement consacré pratiquement 4 jours pleins à AngularJS. Comme mentionné précédemment, ce cadriciel étant à la fois complexe et structurant, il fera l'objet d'une attention toute particulière dans la suite de ce document.

1.4.4 POINTS DE BLOCAGE

Il est évidemment illusoire d'espérer qu'un logiciel soit parfait à la fin d'une période d'intervention, mais certains problèmes doivent tout de même être mentionnés.

- Tout d'abord, la communication via WebSockets, c'est-à-dire toute la partie permettant à un élève d'exécuter un exercice, ne fonctionnait déjà plus lorsque je suis arrivé en mars 2015, alors que cela fonctionnait au moment du départ de mon prédécesseur. L'élément déclencheur semble être une intervention extérieure sur le serveur FORMID et son redémarrage. La cause profonde reste inconnue.
- Ensuite, le serveur de praxéologies était arrêté. La partie assistance de l'auteur grâce à la modélisation des connaissances didactiques ne fonctionnait donc plus. C'est un serveur qui n'est pas géré par les personnes de FORMID, mais dépendant tout de même de l'équipe MeTAH. J'en parlerai dans la partie 2.1.5 qui expose les événements notables à ce propos.

Ces problèmes m'ont accompagné pendant de longs mois. L'ensemble des événements y afférant sont parallèles dans le temps mais comme ils sont indépendants, leur présentation sera séparée.

1.5 STRATÉGIE

Compte tenu de tous les éléments en ma possession après la phase de lancement, j'en conclus plusieurs choses.

La gestion de projet me paraît envisageable, je vais donc consacrer une partie de mon temps à cela. Ensuite, compte tenu de ma présence de longue durée sur ce projet, je choisis de m'arrêter pour appréhender chaque concept nouveau, au lieu d'essayer de survivre dans un code que je ne maîtrise pas. Cela est significatif au vu du nombre de composants, tant côté client que serveur.

Je décide également de créer des documentations, tout d'abord parce que j'ai l'habitude de documenter mon travail, pour moi et pour les suivants, mais aussi parce que je considère que cela fait partie d'un travail bien fait. De plus, lorsque je constate que je n'ai pas les documentations dont j'aurais besoin, je fais en sorte que le suivant ou la suivante ne perde pas de temps sur les mêmes écueils. Je décide également de documenter le code. J'en ai pris l'habitude parce que je considère que quelques explications sur la finalité d'un morceau de code aident énormément à comprendre dans quelle logique il s'inscrit.

Je choisis également de suivre la philosophie initiée par mon prédécesseur tant que je la considère saine et qu'elle ne m'empêche pas d'atteindre mes objectifs. Je n'ai donc surtout pas pour idée de remettre systématiquement en cause ce qui a été fait. Le premier axe suivi est le fait de pérenniser le travail effectué. Cela commence avec l'utilisation du gestionnaire de code source, mais je prévois surtout de l'enrichir par la production de documentations.

Il m'importe de beaucoup communiquer dans ma manière de fonctionner. Je commence par faire des rapports réguliers qui concernent ce que j'ai réalisé, ce que je pense faire ensuite, et mes interrogations, puis je m'adapte aux attentes de mes responsables. Dans le cas présent j'ai été autorisé à communiquer autant que nécessaire par courriel, je n'ai donc pas cherché à limiter ou à regrouper mes envois.

1.6 CONCLUSION DU CHAPITRE

L'immersion dans quelque environnement professionnel que ce soit nécessite beaucoup d'apprentissage et d'adaptation, et à plus forte raison lorsque la technique n'est pas maîtrisée. Après la prise en main des outils, des technologies et du produit, l'assurance s'installe. La capacité opérationnelle se développe alors, et permet de commencer à viser les objectifs. La partie suivante va continuer dans la chronologie et traiter des réalisations importantes qui ont été effectuées dans la suite logicielle FORMID.

2 RÉALISATIONS

Cette partie présente les travaux importants effectués, en détaillant les problématiques et les solutions apportées. Afin de structurer l'ensemble et pour garantir la lisibilité, ce thème sera abordé de deux manières. Tout d'abord par la vue des évolutions fonctionnelles pour l'utilisateur de FORMID, puis par l'approche purement technique de certaines réalisations.

2.1 ASPECTS FONCTIONNELS

2.1.1 CONNEXION À UN DISPOSITIF EXTERNE TANGIBLE

Le dispositif tangible devait permettre à un élève de manipuler des bûchettes, d'éventuellement les regrouper en fagots grâce à des élastiques, et de les placer dans des boîtes. La connaissance par un logiciel du nombre de bûchettes présentes dans chaque boîte sous-entend la création d'une interface homme-machine. Elle a été réalisée par Amiqual4home avec des dispositifs de pesée qui déterminent le nombre de fagots et de bûchettes par la mesure du poids. Le dispositif tangible photographié Figure 14 était déjà fonctionnel de manière autonome. Les mises au point avaient pour la plupart déjà été effectuées, comme le fait d'utiliser des bûchettes en plastique plutôt qu'en bois, pour garantir la constance de leur masse quelle que soit l'hygrométrie. La contrainte d'utilisation est de ne poser qu'un fagot ou un lot de buchettes à la fois.

Figure 14 : ensemble matériel du dispositif tangible.
À gauche, le mini PC Raspberry Pi modèle B avec le système d'exploitation Arch Linux ARM et le logiciel serveur « node.js ».
À droite les balances de pesée construites autour d'un Arduino mini pro, d'un capteur à jauges de contrainte et d'un afficheur.
La communication s'effectue en Bluetooth 4 entre les balances et le Raspberry Pi, et en filaire entre le Raspberry Pi et le réseau local.

J'ai validé la partie communication d'un cahier des charges qui avait été établi conjointement entre MeTAH et Amiqual4home. Le principe est que le dispositif propose une interface REST à laquelle FORMID se connecte pour connaître le contenu des boîtes.

Pour la partie réseau, contacter le mini PC demande de connaître son adresse. Au sein du LIG, une adresse fixe lui est réservée. Par contre, pour la mise à disposition dans les écoles, une solution doit être trouvée. Soit la configuration réseau sera adaptée manuellement, soit nous pouvons envisager du "Multicast DNS" (mDNS) [**MULTICAST DNS15**]. Cette méthode permettrait au mini-PC d'informer le reste du réseau de son nom et de son adresse.

Le programme FORMID communiquait avec les dispositifs externes existants de façon synchrone, ce qui produit un code simple, puisque la réponse est attendue avant d'exécuter l'instruction suivante. Cette solution n'était pas envisageable pour des appels distants à un service REST. Comme un code écrit pour de l'asynchrone sait gérer des appels synchrones, mais pas l'inverse, j'ai dû modifier en profondeur les différents appels génériques au dispositif externe, ainsi que les adaptateurs de communication existants : donc *TPElec* (électricité) et *TPNum* (billes). Les langages de programmation possèdent en général des solutions au problème de gestion de l'asynchronisme (Java, JavaScript et Microsoft C# par exemple en proposent). L'aspect asynchrone, à gérer côté client, donc programmé en JavaScript, a été implémenté grâce à des "promesses" (*Promise* en anglais), qui seront expliquées dans la partie 3.1.

<table>
<tr><th>Cas d'expérimentation tangible</th><th>Données utiles reçues du service REST (au format JSON)</th><th>Variables générées dans FORMID</th></tr>
<tr>
<td> 

Boîte A : 7 bûchettes non liées
Boîte C : 4 bûchettes non liées, un fagot de 15 bûchettes et un fagot de 6 bûchettes</td>
<td><pre>{
   "containers":{
      "A":{
         "nb_units":{
            "value":7,
            "type":"int"
         },
         "nb_bundles":{
            "value":0,
            "type":"int"
         },
         "bundles_content":{
            "value":"",
            "type":"string"
         }
      },
      "C":{
         "nb_units":{
            "value":4,
            "type":"int"
         },
         "nb_bundles":{
            "value":2,
            "type":"int"
         },
         "bundles_content":{
            "value":"15,6",
            "type":"string"
         }
      }
   }
}</pre></td>
<td>
<table>
<tr><th>Nom</th><th>Valeur</th></tr>
<tr><td>A_units</td><td>7</td></tr>
<tr><td>A_bundles</td><td>0</td></tr>
<tr><td>A_weight</td><td>-7.87</td></tr>
<tr><td>A_bundlesContent</td><td></td></tr>
<tr><td>A_bundles_total_units</td><td>0</td></tr>
<tr><td>A_bundles_min_units</td><td>0</td></tr>
<tr><td>A_bundles_max_units</td><td>0</td></tr>
<tr><td>A_total_units</td><td>7</td></tr>
<tr><td>C_units</td><td>4</td></tr>
<tr><td>C_bundles</td><td>2</td></tr>
<tr><td>C_weight</td><td>79.67</td></tr>
<tr><td>C_bundlesContent</td><td>15,6</td></tr>
<tr><td>C_bundles_total_units</td><td>21</td></tr>
<tr><td>C_bundles_min_units</td><td>6</td></tr>
<tr><td>C_bundles_max_units</td><td>15</td></tr>
<tr><td>C_total_units</td><td>25</td></tr>
<tr><td>ALL_units</td><td>11</td></tr>
<tr><td>ALL_bundles</td><td>2</td></tr>
<tr><td>ALL_bundles_total_units</td><td>21</td></tr>
<tr><td>ALL_bundles_min_units</td><td>6</td></tr>
<tr><td>ALL_bundles_max_units</td><td>15</td></tr>
<tr><td>ALL_total_units</td><td>32</td></tr>
</table>
</td>
</tr>
</table>

Figure 15 : étapes de création des variables du dispositif tangible.
Les informations des bûchettes et fagots (à gauche) sont disponibles via un service REST au format JSON (au centre), et servent à créer une liste de variables dans FORMID (à droite). Les variables en caractère gras indiquent juste les dernières modifications ayant eu lieu.
Les variables « weight » sont négligeables car inutilisées, donc éliminées de l'extrait JSON.

Lorsque FORMID contacte le service REST, il obtient des données au format JSON. Vient alors la phase de transformation en une liste de variables. Il m'a paru intéressant d'ajouter des variables calculées pour faciliter l'observation de situations. La Figure 15 montre que deux boîtes fournissent six valeurs qui donnent naissance à vingt variables. Les variables calculées peuvent servir par exemple à vérifier que tous les fagots contiennent le même nombre de bûchettes, grâce aux nombres minimum et maximum de bûchettes par fagot.

Expérimentation en classe

Considérant mon intervention sur FORMID qui arrivait à son terme, mes responsables ont exprimé le désir de pouvoir valider l'utilisabilité du dispositif tangible dans une école.

L'organisation, effectuée par Hamid Chaachoua, a donné lieu à un rendez-vous avec une maîtresse de CE1, et sa classe, en vue d'un test *in situ*. Dans le même temps, Amiqual4home mettait la dernière main à la préparation de deux dispositifs tangibles.

J'avais face à cette situation plusieurs problèmes. Premièrement, les événements étaient très rapprochés, puisque les nouveaux dispositifs tangibles ont été mis à notre disposition le mardi 24 novembre, alors que l'expérimentation était prévue pour le jeudi 26 novembre. Deuxièmement, FORMID ne gérait qu'un seul dispositif tangible à la fois. Et enfin, il fallait trouver une solution pour trouver l'adresse IP d'un mini-PC branché sur un réseau informatique inconnu.

Concernant les deux premiers problèmes, j'ai dû anticiper les développements. Grâce à l'architecture de communication que j'ai précédemment créée pour les dispositifs externes, il a été aisé de proposer à l'utilisateur, auteur ou élève, une liste des dispositifs tangibles. Cependant, cette liste était dans ce cas proposée depuis des paramètres et non depuis une liste de dispositifs réellement actifs. Concernant le dernier problème, la reconnaissance en réseau, j'avais demandé à Amiqual4home d'installer le protocole mDNS sur les mini-PC.

Figure 16 : expérimentation en classe de CE1. Les élèves ont travaillé sur deux postes en parallèle.

Malgré les délais très serrés, l'expérimentation en classe a pu être mise en place (Cf. Figure 16). Les dispositifs tangibles ont été trouvés par leur nom grâce au protocole mDNS, et le scénario élaboré par Anne Lejeune s'est bien comporté.

Les 11 élèves ont donc pu mener à bien l'exercice consistant à regrouper des fagots de 10 bûchettes dans le contenant de gauche pour représenter les dizaines, et des bûchettes seules dans le contenant de droite pour représenter les unités.

J'ai pu tirer des enseignements de cette mise en situation. En premier lieu, chaque élève était accompagné d'un adulte chargé d'observer, de guider si nécessaire, et de détecter les erreurs de comptage des bûchettes par le dispositif tangible. En effet, une contrainte liée au matériel utilisé est d'attendre 2 secondes entre deux actions. Or cette attente n'est pas du tout évidente à respecter lors de la manipulation d'objets réels, et d'autant plus pour des élèves qui doivent réfléchir à solutionner l'exercice.

D'autre part, le protocole mDNS a fonctionné, mais je n'ai pas réussi à déterminer formellement pourquoi. En effet, le principe exige que l'émetteur et le récepteur gèrent ce protocole. Les mini-PC ont été reconnus tout de suite depuis Windows, malgré le fait que ce système d'exploitation ne gère pas nativement ce protocole. J'émets donc les hypothèses suivantes : soit l'un des logiciels installés sur l'ordinateur gère ce protocole, par exemple Mozilla FireFox, soit c'est le serveur DNS du réseau local qui le gère et ajoute les mini-PC dans son annuaire. L'éclaircissement de cet aspect demanderait des recherches plus approfondies.

En conclusion, FORMID a très bien fonctionné et les seules limites ont été celles imposées par la conception du dispositif tangible.

2.1.2 AMÉLIORATION DE LA GESTION DES DISPOSITIFS EXTERNES

La gestion des dispositifs externes est allée bien au-delà de ce qui était initialement envisagé. Une grande attention et beaucoup de jours ont été consacrés au sujet. Les voies explorées sont fructueuses et ouvrent de nouvelles perspectives à FORMID.

Ces évolutions seront très largement détaillées dans la partie technique qui suit. D'un point de vue purement fonctionnel, les évolutions ont été multiples. Tout d'abord la sauvegarde des circuits électriques de *TPElec* ne pose plus de problèmes de gestion des circuits. Ensuite, les micromondes sont maintenant gérés comme les simulations concernant l'affectation de leur état au niveau de l'étape. Ensuite, FORMID accueille maintenant cinq dispositifs externes hétérogènes : le micromonde électrique *TPElec* dans un composant *Adobe Flash*, deux simulations *TPNum* et *LettersGame* en HTML, le dispositif *Buchettes* tangible, et la simulation *SimuBûchettes* sur tablette tactile Android. Et de manière générique, les variables du dispositif externe connecté sont mises à jour en permanence dans l'interface FORMID, pour pouvoir observer l'effet d'une action dans le dispositif externe sur celles-ci. L'ensemble de ces évolutions ouvre vraiment de nouvelles perspectives à FORMID.

2.1.3 EVOLUTION DU MODULE AUTEUR

Le module auteur a d'une part beaucoup évolué par tout petits incréments, et d'autre part subi des améliorations fonctionnelles importantes.

La Figure 17 et la Figure 18 présentent certaines différences qui ne servent que de support aux explications qui vont suivre, puisque le fonctionnement n'est pas saisissable par une photo.

Figure 17 : ancienne vue du module auteur.
Les boutons du haut sont toujours affichés mais activés ou non. À droite en gris le bouton permettant d'importer les variables du dispositif externe vers la partie « Variables simulation ». Les parties "Variables simulation", "Simulation", "Ontologie" peuvent se plier/déplier.

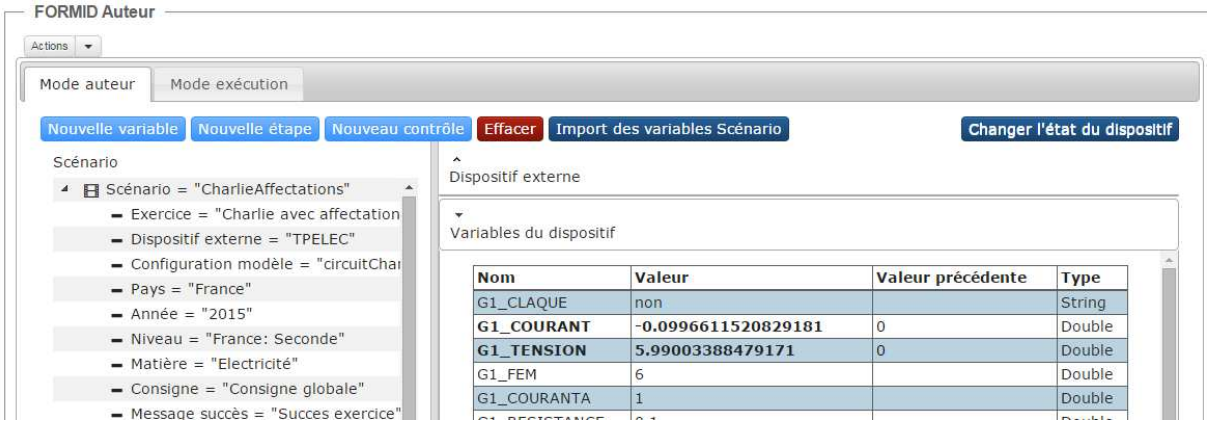


Figure 18 : nouvelle vue du module auteur.
Les boutons du haut ne s'affichent que s'ils sont utilisables. Les variables sont celles du dispositif, rafraichies en temps réel.
Les parties "Dispositif externe", "Variables du dispositif" peuvent se plier/déplier.

Résolution d'anomalies

Les modules élève et tuteur étaient bien terminés et fonctionnels, ce qui n'était pas le cas du module auteur. D'une part certaines fonctions étaient absentes, comme le fait de pouvoir rouvrir un scénario sauvegardé ou publié. Et d'autre part de nombreuses anomalies émaillaient son fonctionnement. J'ai donc rendu le code bien plus résistant pour qu'il ne cède pas lorsque la liste des variables du scénario est vide, par exemple. J'ai également œuvré pour que les erreurs produites côté serveur affichent un message à l'utilisateur.

Amélioration de la mise en page et de l'ergonomie

Un problème récurrent se produit lors du redimensionnement de l'écran. Cet aspect invisible a été géré pour que le dispositif externe affiché dans l'onglet "Simulation" utilise bien tout l'espace disponible, même lorsque l'écran change de taille. Des boutons ont été déplacés, la simulation placée en haut, et les boutons habituellement grisés ne s'affichent que lorsque le contexte s'y prête. J'ai donc progressivement adapté l'interface, et en l'absence de retour négatif, les différentes propositions ergonomiques sont restées.

Gestion des variables

L'ancien fonctionnement était ainsi : le tableau présent dans le groupe "Variables simulation" devait être peuplé par le bouton gris "Importer" visible à droite de la Figure 17. Puis l'auteur sélectionnait les variables qu'il voulait gérer dans le scénario grâce à des cases à cocher et cliquait sur "Import variables sélectionnées". Ce processus me paraissait contre-intuitif. De plus, nous rencontrions des problèmes de variables en doublon.

Le rôle du bloc des variables a changé. Maintenant, il affiche en temps réel toutes les variables fournies par l'adaptateur de communication du dispositif externe. Les dernières variables dont la valeur a changé s'affichent en gras. L'auteur peut donc visualiser la modification induite par chaque action, et concevoir son scénario en conséquence. Par contre il n'utilise plus cette zone pour choisir les variables à intégrer dans le scénario. Il a un nouveau bouton "Import des variables Scénario" Figure 18. Ceci ouvre une fenêtre de choix reproduite Figure 19. Elle permet de choisir les variables avant d'importer et même de supprimer les variables devenues inutiles.

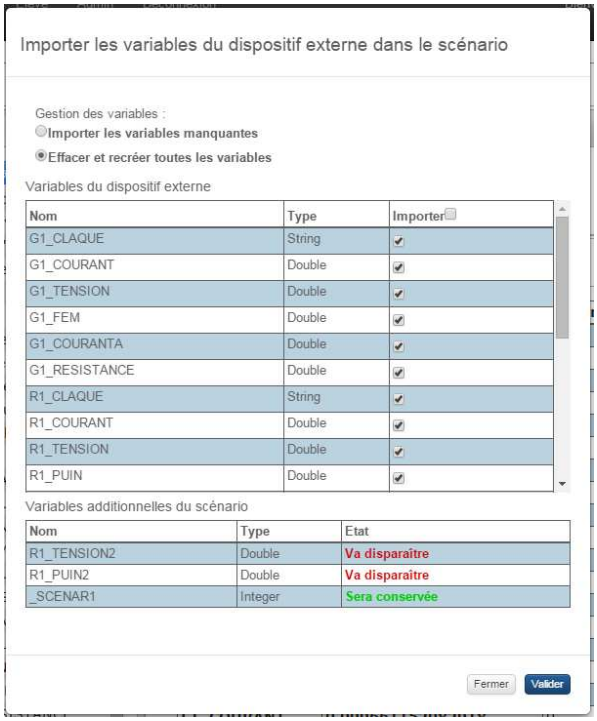


Figure 19 : écran de choix des variables du dispositif externe à importer dans le scénario.

Mes responsables m'ayant fait part de leur désir d'avoir des « variables scénario », j'ai développé comme prototype la possibilité d'ajouter manuellement dans le scénario une variable commençant par un signe particulier (« _ ») que l'auteur pourrait manipuler comme bon lui semble. Il est dorénavant possible grâce à cela de, par exemple, déclencher un contrôle si une valeur calculée augmente ou diminue (le nombre de lampes simultanément allumées par exemple).

J'ai ensuite offert une autre fonctionnalité à éprouver, la possibilité d'intégrer des valeurs de variables dans les messages envoyés à l'élève, ce qui permet par exemple d'afficher « *vous avez grillé 2 fois la lampe* ». Cette possibilité a été approuvée et a donc été étendue à tous les messages et consignes.

Nouvel assistant de contrôle d'une situation observable.

Ces contrôles se déclenchent automatiquement pendant la manipulation d'un dispositif externe par l'élève, et visent à analyser sa progression, en détectant une potentielle erreur commune ou un passage obligatoire vers la résolution.

En interne, le mécanisme régissant un contrôle de situation observable et celui de la validation d'une étape par l'élève est le même. Un contrôle, donc, est un morceau de code Java qui évalue des variables du dispositif externe pour renvoyer une valeur booléenne (oui ou non) indiquant s'il se déclenche ou non. Or, ce code Java est complexe à saisir pour des non-initiés. Il était déjà possible de faire une photographie de la situation actuelle comme condition de déclenchement, après avoir choisi les variables pertinentes. La Figure 20 montre le nouvel écran qui s'affiche lorsque l'auteur souhaite effectuer une photographie. Chaque variable du dispositif externe gérée dans le scénario s'affiche, avec sa valeur courante. Il suffit à l'auteur de cocher les lignes à considérer, pour chacune de choisir un opérateur (égal, différent, supérieur, contient, etc.) et d'éventuellement modifier la valeur avant de valider, pour générer le code Java.

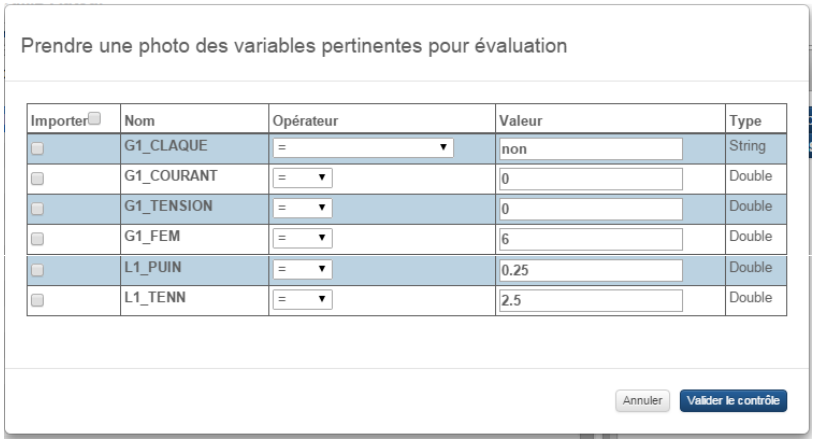


Figure 20 : écran de choix lors de la photographie "de contrôle" du dispositif externe.

L'assistant couvre uniquement les cas simples habituels, c'est pourquoi le code Java est toujours utile. Néanmoins, il est plus facile d'en comprendre la syntaxe après génération que de partir d'une feuille vierge. Et à ce propos, j'ai créé une liste de méthodes utilitaires pour uniformiser ce que voit l'auteur. Par exemple, il verra « `areEqual(G1_FEM, 6)` », cette méthode `areEqual` se comprend aisément, et a été créée pour tous les types de données. Et dans le cas d'un type décimal, une tolérance est appliquée à la comparaison.

Bien entendu, nous restons dans de la programmation, mais pour des comparaisons basiques cela semble assez accessible. Pour faciliter encore la saisie de ce code, j'ai implémenté une coloration syntaxique. De la sorte, les commentaires, méthodes, opérateurs ont chacun leur couleur. De plus, en plaçant le curseur sur une parenthèse ouvrante, la parenthèse fermante correspondante se met en surbrillance. Cela facilite grandement la mise au point du « programme » codant pour la condition à évaluer.

Mode exécution pour l'auteur

Cette partie était entièrement structurée tant du côté client que du côté serveur, mais beaucoup de problèmes restaient à résoudre et de mises au point à effectuer.

Le défi le plus complexe consistait à faire cohabiter deux fois le même dispositif externe dans la même page web. Car même si visuellement un seul des deux onglets "Mode auteur" et "Mode exécution" n'est visible à la fois, d'un point de vue page web, ils sont côte à côte. C'est-à-dire que le fichier HTML spécifique à la simulation est importé deux fois. Comment permettre de gérer deux fois le même code source, et deux fois les mêmes identifiants d'éléments, telles ont été mes grandes questions. Ma solution a été de transformer les pages spécifiques aux adaptateurs de communication pour qu'elles soient des pages JSP (« JavaServer Pages »), donc dynamiques, et qu'elles prennent en paramètre le mode "auteur" ou "élève", qu'elles répercutent sur les éléments mentionnés. J'ai dû gérer l'interrogation automatique du dispositif externe : puisque deux composants sont présents, les requêtes se font en double. Cette partie a été bien plus compliquée que prévu puisque l'affichage est géré par le cadriciel AngularJS.

Le mode exécution se comporte donc comme un mode élève, excepté le fait qu'il n'est nullement nécessaire de sauvegarder, compiler ou publier le scénario en cours d'élaboration, le travail est masqué. L'autre différence concerne le fait qu'aucune trace élève n'est enregistrée, puisque l'exercice est virtuel et qu'il n'y a pas de séance, entre autres raisons.

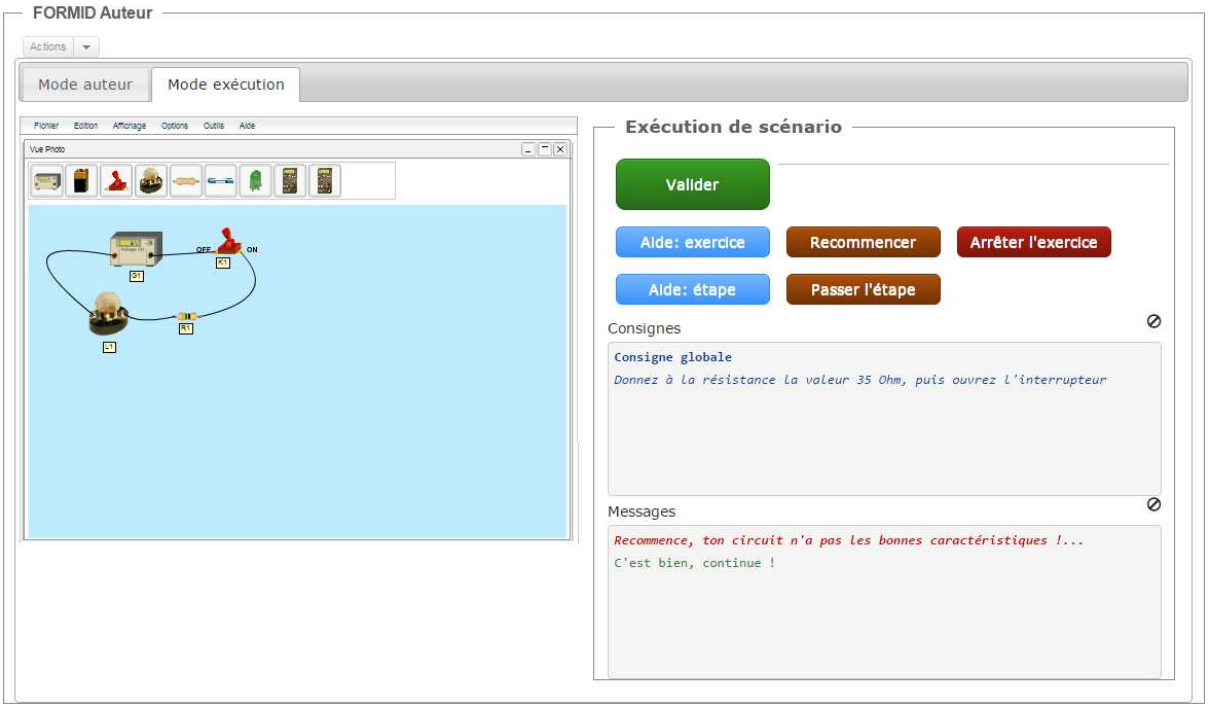


Figure 21 : le mode exécution du module auteur se comporte comme le module élève.

Finalement, le mode exécution est devenu très propre et stable grâce à l'incorporation des adaptateurs de communication dans des contrôleurs du cadriciel AngularJS.

Assistance à la saisie

Dans ce que l'on appelle l'arbre du scénario, à gauche de la Figure 9 (page 10), bon nombre de feuilles peuvent être modifiées. Ce que l'on appelle feuille, ce sont les données qui ne peuvent pas être déployées pour en afficher d'autres, comme par exemple le nom d'une variable, la consigne de l'exercice, la valeur en points donnée à une étape, ou les contrôles de situations observables que l'on souhaite activer pour une étape.

Lors du double clic sur une feuille pour en modifier le contenu, le code qui était en place affichait un écran de saisie parmi trois, suivant le type de donnée à saisir : une chaîne de caractères, une grande zone de texte ou une valeur booléenne (oui/non).

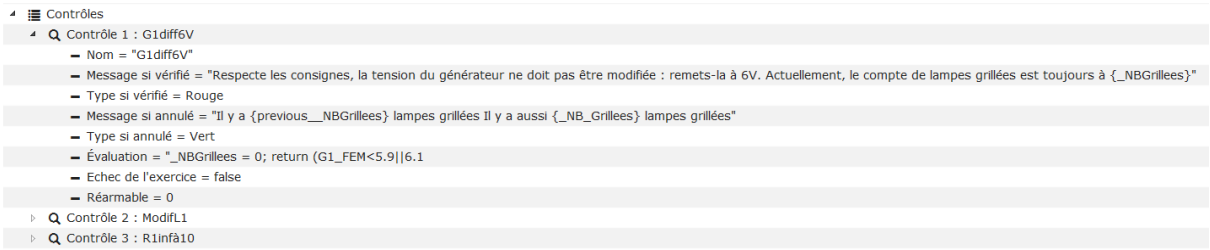


Figure 22: contrôle de situation observable extrait de l'arbre de scénario.

Peu à peu j'ai enrichi la saisie de nouveaux écrans pour faciliter l'usage du module auteur. Par exemple dans la Figure 22 la valeur « Type si vérifié » avait une valeur « 1 », sans explication de sa signification ni de liste des valeurs possibles. La Figure 23 montre le nouvel écran de choix, qui présente uniquement une couleur.

Figure 23 : écran de choix du type de message si un contrôle se déclenche.

Les écrans de saisie sont finalement au nombre de dix, dont font partie la saisie du code Java ainsi que le choix des contrôles à déclencher dans une étape.

J'ai également ajouté des messages d'information sous différentes zones de saisie pour guider ou rappeler un fonctionnement qui ne peut pas être deviné. Par exemple sous la saisie de la consigne de l'étape, l'auteur voit le message Figure 24.

Figure 24 : exemple d'information affichée sous une zone de saisie du module auteur.

Toujours dans l'optique d'aider l'auteur, l'arbre du scénario contient des infobulles calculées. C'est-à-dire qu'en passant par exemple la souris sur le nœud contenant toutes les variables, nous voyons la liste des variables sans même ouvrir le nœud. Cet accès facilité à l'information a été développé suite au constat que nous passions énormément de temps à ouvrir et fermer des nœuds.

Contrôles réarmables

Un contrôle se déclenche si la situation observable est réalisée. Lors du déclenchement, il affiche un message à l'élève, dans la couleur désirée. Puis, tant que la situation ne disparaît pas, le contrôle reste enclenché et n'affiche plus de message. Cela m'a posé un problème lorsque j'ai voulu concevoir un scénario qui affichait « vous progressez » à chaque fois que le nombre d'éléments corrects augmentait. Le fait de progresser trois fois de suite ne déclenche qu'une fois le contrôle. J'ai donc introduit cette notion de « réarmable » pour qu'il soit possible de déclencher le contrôle un nombre de fois spécifié même si la condition initiale n'a pas disparu.

Autres améliorations

De nombreuses autres améliorations ont été intégrées au module auteur, mais étant mineures elles ne représentent pas grand intérêt à être listées.

2.1.4 ÉVOLUTIONS DU MODULE ÉLÈVE

Les évolutions du mode élève sont hétéroclites, en voici une vue d'ensemble.

Moyens d'action

Il est maintenant possible pour l'élève de recommencer l'étape courante. Le dispositif externe se place alors automatiquement dans l'état dans lequel il était au début de l'étape, par exemple le circuit électrique reprend la forme exacte qu'il avait. Évidemment tous les dispositifs externes ne le permettent pas, notamment le dispositif tangible pour lequel il faudrait une intervention physique. Si l'élève souhaite passer une étape, le dispositif peut également se placer dans le bon état. Les rouages techniques de la gestion des états sont décrits dans la partie 2.2.2.

Ergonomie

Dans le but de faciliter l'accès et la compréhension, les boutons d'action ("Valider", "Recommencer", etc.) ont été modifiés en taille, en couleur, en disposition et en libellé. Leur apparence finale a été illustrée Figure 10 (page 11).

Les zones d'affichage des consignes et des rétroactions sont maintenant nettoyées automatiquement pour faciliter la lecture. Et les nouveaux messages sont animés lors de leur affichage pour les rendre bien visibles.

Synthèse vocale

Toute une partie des scénarios pédagogiques élaborés avec le dispositif tangible est destinée à des élèves de CP/CE1. Or, dans ces niveaux, les élèves ne savent pas forcément lire, ou ne sont pas encore en mesure de comprendre une consigne un peu longue. Nous nous faisions souvent la remarque du fait que l'élève dans ce contexte ne peut pas utiliser seul FORMID.

J'ai donc voulu tester l'ajout d'une synthèse vocale à FORMID. Il a été très rapide de trouver des composants JavaScript. J'ai testé, MeSpeak [**MESPEAK15**] qui a rapidement fonctionné, mais qui souffre d'une qualité médiocre. J'ai cherché à nouveau et trouvé ResponsiveVoice [**RESPONSIVEVOICE.JS15**] de qualité bien supérieure. Je l'ai intégré à FORMID, et depuis, les consignes sont lues en plus d'être affichées.

Ce que je trouve remarquable avec cette synthèse vocale, c'est que cela ouvre une dimension nouvelle à FORMID en supprimant une bonne partie de la problématique de la lecture pour les jeunes enfants, alors qu'au total cela ne m'a pris que 2 heures de mon temps.

2.1.5 DIDACTIQUE ET PRAXÉOLOGIES

2.1.5.1 Théorie

Dans une équipe étudiant l'apprentissage humain, la didactique est le cœur de métier. Le triangle pédagogique représente la manière dont l'Enseignant va communiquer un Savoir à un Apprenant (triangle EAS). La méthode d'apprentissage doit estomper les liens Apprenant-Enseignant et Enseignant-Savoir et privilégier l'interaction Apprenant-Savoir. Cela nécessite la mise en place de situations pédagogiques appropriées. [**HOUSSAYE00**]

La praxéologie, à la base, est l'analyse de l'action humaine. Un modèle praxéologique qui fait référence a été établi par [**CHEVALLARD**] en 1994, il vise à décrire tous les comportements humains. La Théorie Anthropologique du Didactique, ou TAD, véhicule l'idée selon laquelle toutes les actions humaines peuvent être unifiées pour être analysées de la même manière, quels que soient les processus de pensée qui y ont mené et la culture de ceux qui les réalisent. À cela s'ajoute la notion d'écologie, qui fait référence à l'aspect évolutif des praxéologies, qui naissent, évoluent et disparaissent.

Le modèle praxéologique part de la définition selon laquelle "*toute activité humaine consiste à accomplir une tâche t d'un certain type T, au moyen d'une technique* τ *(tau), justifié par une technologie* θ *(thêta) qui permet en même temps de la penser, voire de la produire, et qui à son tour est justifiable par une théorie* Θ *(thêta majuscule)* [**CHAACHOUA14**]. Les quatre "t" de la théorie et « TEL » issu de « *Technology-Enhanced Learning* » ont donné en 2010 naissance au projet T4TEL actuellement en cours dans le laboratoire MeTAH.

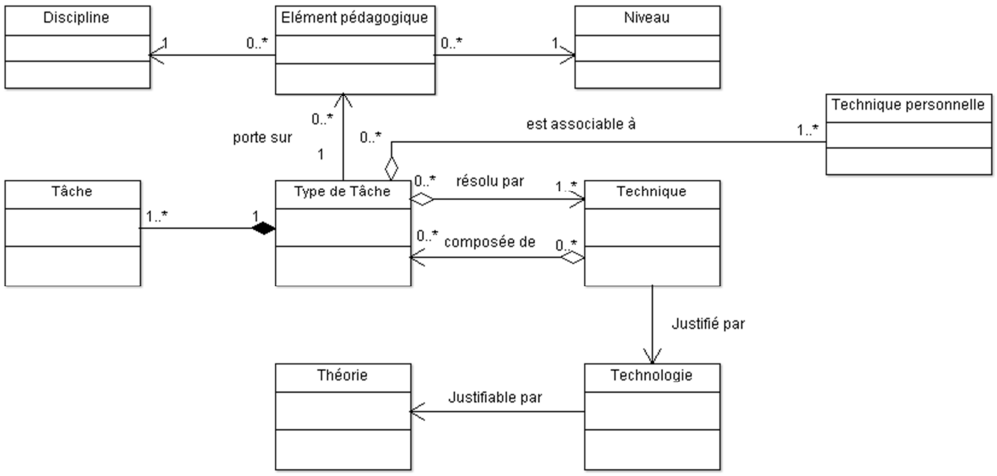


Figure 25 : modèle praxéologique.
Le cœur du modèle est constitué du type de tâche et de la technique. Le type de tâche est résolu par une ou plusieurs techniques. Chaque technique peut être composée de types de tâches à effectuer pour la réaliser.
Source : modèle conçu à partir d'explications d'Anne Lejeune et de [**CHAACHOUA14**].

Le modèle praxéologique, représenté Figure 25, est également enrichi de ce que l'on appelle les "praxéologies personnelles". Elles représentent les comportements a priori d'apprenants, qui suivent une technique qui peut être correcte, mais aussi potentiellement erronée ou non appropriée pour la résolution d'une tâche. L'aspect de la connaissance des erreurs courantes est très important pour guider l'apprentissage.

Concrètement pour FORMID, qui est un outil d'élaboration de scénarios pédagogiques, l'ambition est de s'appuyer sur des modèles praxéologiques établis par domaine (mathématiques, biologie, chimie, etc.), par institution et par niveau, pour aider le concepteur à placer l'apprenant dans des situations pertinentes. Ils devraient également permettre de déterminer quels sont les observables incontournables des situations où les risques d'erreur sont identifiés.

2.1.5.2 Étapes importantes

Les praxéologies en général ont donné naissance au logiciel MeTAH « Ontoprax » permettant leur saisie. Ce logiciel stocke les praxéologies sous forme d'ontologies. En introduction sur le sujet, les ontologies sont une représentation d'une partie du monde réel et décrivent très précisément des entités et des relations. Un des avantages indéniable à l'utilisation d'ontologies est la disponibilité de moteurs d'inférence qui peuvent déduire des règles à partir d'autres règles logiques. Par exemple si des bûchettes et des fagots sont des éléments, et si une boîte peut contenir des éléments, et que A est une boîte, alors A peut contenir des bûchettes et des fagots. Dans le domaine de la praxéologie, c'est un sujet en cours de définition. Les concepteurs y trouvent l'avantage de ne pas devoir écrire toutes les règles logiques atomiques.

Le défaut de fonctionnement du serveur de praxéologies a été un problème omniprésent, face à un besoin réel. C'est-à-dire qu'il n'y a jamais eu d'échéance fixe, mais comme le sujet revenait en permanence il a été pris la décision de commencer des fac-similés, puis de les enrichir jusqu'à simuler un serveur de praxéologies fonctionnel.

2.1.5.3 Fac-similé de praxéologies

Créer un fac-similé du serveur de praxéologies a été un réel défi.

Pendant les quelques heures où le réel serveur de praxéologies avait fonctionné, j'avais eu l'idée de sauvegarder immédiatement dans un fichier local la réponse JSON obtenue. Je me suis vite aperçu en créant le faux serveur que ces données sauvegardées définissaient seulement un cadre général qui déclenchait de multiples autres requêtes pour obtenir le détail des types de tâches, techniques, mauvaises pratiques et variables. Et malheureusement je n'avais aucun exemple de ces fichiers. Un document de mon prédécesseur contient le modèle de praxéologies utilisé et quelques copies d'écran de la partie visible. Après analyse au sein de l'équipe, il s'est avéré que le modèle de praxéologies utilisé était devenu obsolète.

Pour synthétiser, je devais générer des données permettant d'obtenir l'illusion d'un serveur fonctionnel sans avoir un exemple de toutes les données réelles, et sans connaître la répartition des éléments observés dans le modèle d'origine. De plus, je ne savais pas quel était le fonctionnement attendu dans FORMID.

La première phase a été de la rétro-ingénierie à partir de traces. C'est-à-dire que suite à l'observation de toutes les requêtes ne pouvant aboutir, j'analysais le code qui devait traiter les résultats attendus, pour créer manuellement un fichier d'exemples qui serait accepté par ce code. Cela a donné lieu à beaucoup d'incertitudes et d'inconnues. Il arrive souvent que l'on doive développer un travail sans avoir tous les éléments, mais il est rare, heureusement, de devoir simuler des données que l'on ne peut même pas observer pour que le programme fonctionne sans connaître le comportement attendu.

La deuxième phase a été l'apport d'informations de la part d'Anne Lejeune et de Viviane Guéraud, sous la forme d'une praxéologie de référence présentée en Annexe 3. Il paraît utile de préciser que celle-ci est un exemple, pour un type de tâches particulier dans un domaine spécifique. Cela m'a permis de comprendre comment les données à plat pouvaient s'articuler, et avec quelles valeurs vraisemblables je devais renseigner mes réponses de fac-similé. La tâche est devenue tout à coup bien plus facile.

2.1.5.4 Développement d'un serveur de praxéologies

Grâce en partie à l'avancée des travaux de MeTAH sur le modèle T4TEL, mes responsables ont travaillé sur l'aspect théorique des informations praxéologiques. Elles m'ont ainsi fourni un lot de données accompagnées de demandes d'évolution. Nous pouvions faire évoluer simplement les données du fac-similé, qui ne représentent pas un vrai modèle, mais de simples réponses au format JSON préenregistrées pour chaque requête. Après de rapides propositions de part et d'autre, Anne Lejeune s'est montrée intéressée par le développement de notre propre serveur avec un vrai modèle de données, et j'ai estimé que cela était possible dans le temps imparti.

C'est ainsi que nous, je vais expliquer le « nous », avons commencé le développement de notre propre serveur de praxéologies à trois semaines de la fin de mon stage. J'ai proposé quelques solutions, et retenu le langage PHP qui est très simple, commun, et qui pouvait donc être mis en œuvre rapidement tant du point de vue du codage que de la préparation des environnements de développement et de production. Côté base de données, une nouvelle base « `Praxeologies` » a été créée sur le serveur PostgreSQL qui héberge FORMID. Afin d'avancer au plus vite, j'ai créé un modèle minimaliste basé sur les données qu'il devait contenir. Je n'ai pas eu le temps de m'en servir, Anne Lejeune m'a fourni un modèle de données complet, rempli, et correspondant à la théorie réfléchie avec Viviane Guéraud. Elle m'a également fourni un lot de fonctions SQL à utiliser pour chaque besoin, avec les explications afférentes. J'avais donc une solution clés en main, et il a été extrêmement rapide de développer le service PHP fournissant des données au format JSON. J'ai donc pu me consacrer aux évolutions de FORMID inhérentes aux nouveautés du modèle de données, et plus précisément du module auteur. Ces développements ont demandé de multiples mises au point puisque modifier l'arbre de scénario peut engendrer des anomalies conséquentes.

La Figure 26 montre un exemple de données praxéologiques. À chaque tâche type correspondent des connaissances mises en œuvre, une méthode de résolution (constituée de tâches type), une liste d'erreurs courantes et d'éventuels facteurs d'influence.

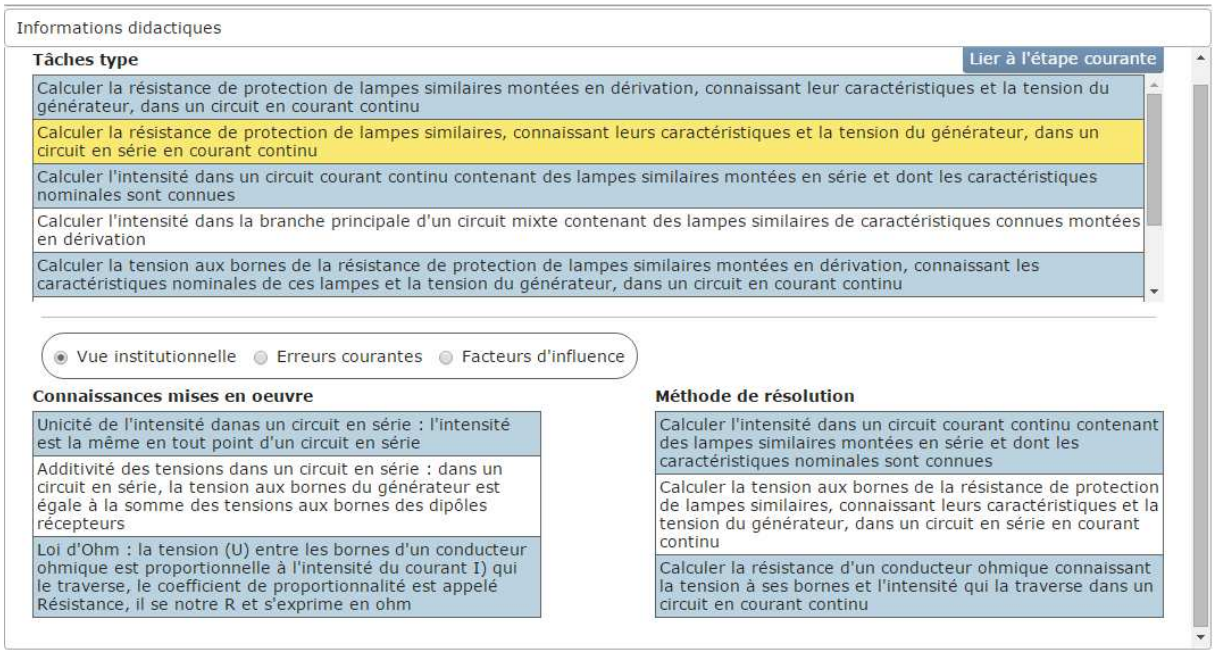


Figure 26 : extrait de données praxéologiques pour l'électricité en quatrième, en France

Un type de tâche peut être associé à chaque étape du scénario, ce qui a pour effet de copier les données praxéologiques dans le scénario : connaissances (loi d'Ohm, de Kirchoff…), méthode de résolution attendue (liste récursive de tâches type à réaliser), erreurs courantes et facteurs d'influence.

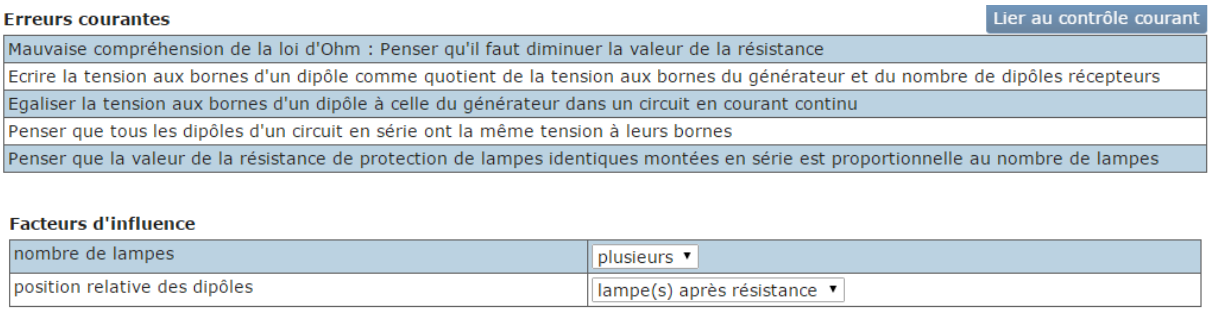


Figure 27 : erreurs courantes et facteurs d'influence de la tâche type sélectionnée Figure 26.

La Figure 27 montre les détails du type de tâche « *calculer la résistance de lampes similaires, connaissant leurs caractéristiques et la tension du générateur, dans un circuit en série en courant continu* ». Une des erreurs de résolution courante est de « *penser que tous les dipôles d'un circuit en série ont la même tension à leurs bornes* ». L'auteur peut donc créer un contrôle spécifique et lui associer cette erreur courante (Cf. Figure 28). Comme un contrôle donné est utilisable dans toute étape, nous enregistrons tous les types de tâches qui amènent potentiellement un élève vers cette erreur courante.

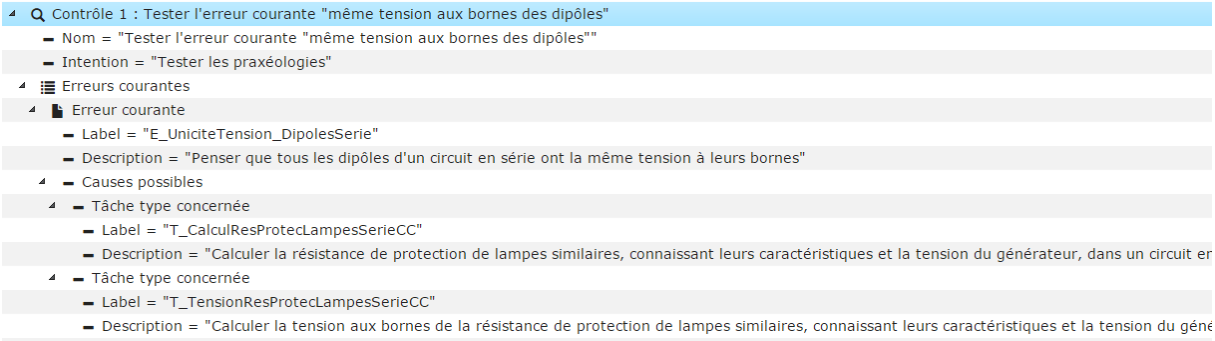


Figure 28 : le contrôle de situation observable est associé à une erreur courante issue du modèle praxéologique

L'affectation de types de tâches et d'erreurs courantes au scénario lui donne une sémantique nouvelle. L'objectif suivant est d'être capable de fournir ces informations au tuteur, mais ce point n'a pas pu être abordé faute de temps.

Les praxéologies sont très importantes pour l'avenir de FORMID. Je pense qu'il est donc parfait qu'Anne Lejeune ait elle-même élaboré le modèle et créé les fonctions que j'ai utilisées. D'une part elle maîtrise parfaitement le modèle de données, et d'autre part comme le serveur que j'ai développé en PHP ne contient pas de commandes SQL, le modèle est vraiment découplé du code. Cette situation est idéale pour les évolutions.

La situation verrouillée depuis plus de 8 mois concernant les praxéologies a ainsi pu être débloquée à une semaine de mon départ. Mais la présente situation ne constitue que le point de départ de certains des objectifs qui avaient été fixés dans le cadre de ce mémoire.

2.2 EVOLUTIONS TECHNIQUES DU LOGICIEL

2.2.1 MISE EN PAGE

S'il est bien un sujet à la complexité sous-évaluée, c'est celui concernant la mise en page en HTML. Des contraintes énoncées clairement peuvent se révéler très complexes à résoudre : par exemple, comment utiliser toute la hauteur disponible, ou comment faire en sorte qu'un élément fasse la même hauteur qu'un autre, ou comment interdire à un élément de passer à la ligne si la largeur de l'écran n'est pas suffisante. En ajoutant à cela des composants de mise en page comme Bootstrap, la mise au point peut devenir une activité très coûteuse.

Les interventions que j'ai effectuées pour gérer la mise en page sont multiples, voici les principales :

1. Mise en page de l'application dans sa globalité : lorsque la largeur du navigateur diminuait, une partie de la vue principale passait derrière la barre de menu horizontale.
2. Dans le module auteur, la réduction de taille provoquait un affichage inutilisable avec de gros éléments qui se chevauchaient.
3. Dans le module élève, le redimensionnement provoquait également de gros problèmes quant à la taille de la simulation et celle de la boîte de contrôle contenant notamment "démarrer" et "arrêter".
4. Dans le module tuteur, la réduction de taille scindait le tableau principal, avec en premier les en-têtes des lignes (la liste des élèves) et en dessous les données afférentes (les exercices et leurs étapes).

Je vais détailler ci-dessous la problématique du 4ème point car le temps passé à résoudre ces problèmes de mise en page justifie bien une mise en lumière.

Mise en page du module Tuteur

Les problèmes de mise en page n'apparaissent en général pas au premier abord, mais lorsque l'espace disponible pour afficher le site internet se réduit. La Figure 29 montre que l'écran de tuteur voit le tableau passer en dessous de la liste des élèves. Ensuite, les exercices disparaissent au fur et à mesure que l'espace se réduit. Dans ces conditions le module tuteur est inutilisable.

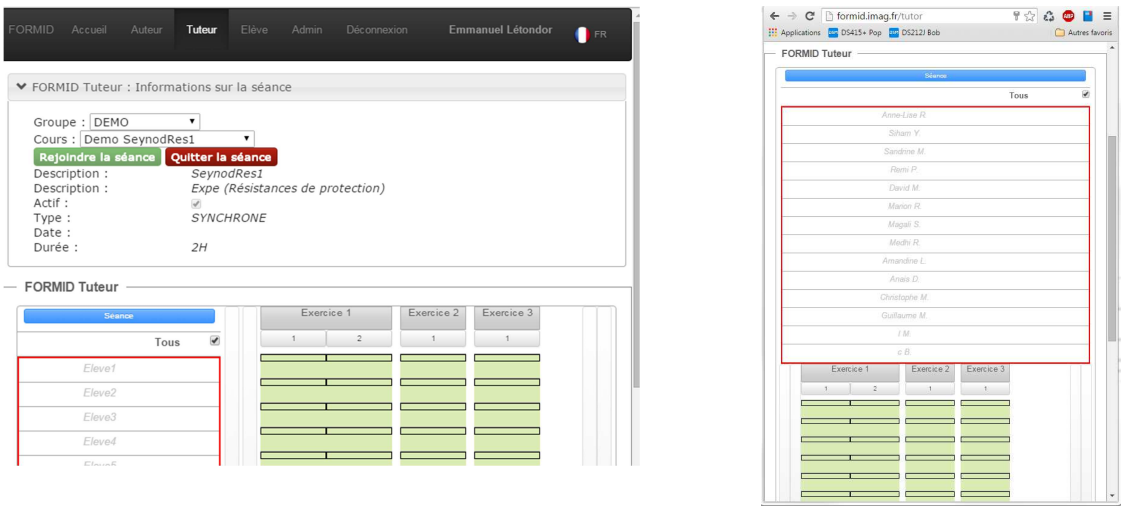


Figure 29 : illustration d'un problème de mise en page.
L'écran du tuteur (à gauche) ne se comporte pas bien en cas de redimensionnement (à droite) : la colonne contenant le détail des exercices passe "à la ligne" et la colonne élève occupe alors toute la largeur disponible. Ce sont des colonnes issues du composant « Bootstrap ».

Il pourrait sembler évident d'ajouter simplement une barre de défilement horizontale pour faire défiler le contenu, au lieu de lui permettre de passer à la ligne. Mais le problème a plusieurs facettes.

En premier lieu, les exercices qui n'ont pas de place dans l'écran ne disparaissent pas sur la droite mais passent "à la ligne" hors du champ visible. Ajouter une barre de défilement (le style CSS est "`overflow:auto`") sur le conteneur ne suffit pas. Il devient alors nécessaire de rechercher dans toute l'arborescence des éléments (pour la plupart des blocs de type "`div`"), pour déterminer quel est leur alignement, s'ils sont en mode flottant, relatif, fixe.

Cette information trouve également sa source dans Bootstrap. Par exemple, une mise en page est composée de colonnes dont la largeur totale fait 12 unités. Avec deux colonnes de rapports 2/12 et 10/12, tout semble fonctionner pour les élèves à gauche et les exercices à droite. Seulement, lorsque l'écran est trop réduit, ce « trop » étant déterminé par Bootstrap, la colonne passe à la ligne. Les styles CSS de version 3 permettent en effet de spécifier des consignes `@media` qui activent des propriétés lorsque la taille de l'écran atteint une taille critique, comme le montre la Figure 30.

```
@media (min-width: 992px) {
  .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6,
  .col-md-7, .col-md-8, .col-md-9, .col-md-10, .col-md-11, .col-md-12 {
    float: left;
  }
```

Figure 30 : extrait de style issu de "bootstrap.css".
Cet extrait montre que différents styles peuvent s'appliquer selon la taille d'affichage, grâce à la directive "@media".
Dans ce cas, les colonnes deviennent des éléments flottants si la largeur totale de l'écran est au moins de 992 pixels.

Cette possibilité est utilisée massivement dans le composant Bootstrap qui a pour ambition de permettre la création de sites internet adaptatifs. Un site adaptatif, en anglais "*Html Responsive Web Design*" [**W3SCHOOLS.COM15**], vise à l'élaboration d'un code unique pour un affichage dans de multiples situations : écrans d'ordinateur, tablettes tactiles, smartphones, donc quelle que soit la résolution de l'écran.

Bien entendu, cet ambitieux objectif est une perspective très intéressante. Dans les faits, l'élaboration est un travail compliqué car une mise en page acceptable ne garantit pas qu'elle le reste quand l'écran changera de taille. À moins de connaître parfaitement les styles CSS et Bootstrap, il est difficile de prévoir qu'une deuxième colonne puisse passer à la ligne sous la première. La mise au point des sites de cette manière n'a plus rien à voir avec une mise en page par tableaux (éléments du style "`<table>`") qui a été beaucoup utilisée du fait de sa simplicité de mise en œuvre, mais qui se révèle incapable de créer des sites adaptatifs.

J'aimerais pouvoir affirmer que ma méthode de résolution est maîtrisée; mais ce n'est malheureusement jamais le cas avec la mise en page en HTML. Malgré ce que la pratique m'enseigne, je continue à tâtonner, à procéder par essais successifs, et à parcourir les forums de discussion.

Résolution du problème du module tuteur

Dans un premier temps il m'a fallu comprendre que le tableau passait sous la liste des élèves parce que les colonnes Bootstrap sont en mode flottant aligné à gauche (de style "`float:left;`"), et que ce mode permet justement ce genre de comportement. Modifier le style de manière locale pour désactiver juste le mode flottant n'a pas fonctionné. Je n'utilise donc plus les styles de colonne Bootstrap, et le comportement indésirable a disparu. Cependant, en réduisant la largeur, les exercices disparaissent un par un. Les blocs qui les composent suivent le même comportement de passage "à la ligne".

Sans entrer dans le détail de l'explication du HTML et du CSS, l'architecture des mises en page récentes utilise surtout les notions ci-après dont la présentation me sert à illustrer les possibilités combinatoires :

- Le style "`float`" pour rendre un élément flottant, qui prend par exemple comme valeur : "`left`" (gauche), "`right`" (droite), qui se couple avec le style "`clear`" permettant de rendre indépendant le reste du flux d'affichage du dernier élément flottant. Un élément flottant possède un comportement particulier : alors que plusieurs éléments flottants se placent côte à côte, si la largeur de la page est insuffisante, ils peuvent revenir à la ligne.
- En HTML il existe des éléments en ligne ("`inline`") comme le "`<span>`", et en bloc ("`block`"), comme le "`<div>`". En principe, un élément en ligne affichera son contenu sur une seule ligne, et un bloc sera en dessous du précédent. Mais le style "`display`" permet de changer le type d'un élément, voire de combiner les caractéristiques avec "`inline-block`". Il existe aussi des valeurs "`table`" et "`table-cell`" permettant de simuler l'affichage d'un tableau avec d'autres éléments.
- Pour afficher une barre de défilement si le contenu est trop grand, il suffit en principe d'appliquer le style "`overflow:auto;`". Mais il faut pour cela que le contenu accepte de dépasser la taille de son contenant, ce qui n'est pas le cas si les éléments sont en mode flottant.

Sachant que les éléments "`<div>`" peuvent être combinés et imbriqués à volonté, nous constatons qu'il peut exister de très nombreuses possibilités avec ce seul échantillon des fonctionnalités présentées de CSS.

Quand un tel cas se présente, il m'arrive d'arrêter le développement, et de créer une page de test indépendante. Sortir du contexte ne garantit pas que la solution trouvée soit applicable, mais permet au moins d'observer le comportement hors de toute dépendance. Aidé de nombreux articles expliquant la mise en page HTML avec du CSS comme [**ALSACREATIONS15**], j'obtiens une maquette du résultat attendu. L'annexe 4 montre la maquette réalisée pour corriger ce problème.

Simuler un comportement de tableau à partir d'autres éléments me donne vraiment l'impression que cette méthode permet juste d'éviter les éléments "`<table>`" qui deviennent tabous pour les créateurs perfectionnistes de sites web. La transposition dans FORMID n'est pas offerte, et malgré des adaptations sur les éléments contenant le tableau, je ne parviens pas au bon résultat. En cause la nécessité d'imbriquer des tableaux.

J'utilise au quotidien le site internet *StackOverflow.com* qui est une mine d'or pour le développeur. Après la lecture de multiples articles, j'en suis parvenu à la conclusion suivante : je vais utiliser des tableaux plutôt que des styles CSS qui les simulent. Même ainsi, il s'est avéré qu'un conteneur de type bloc "`<div>`" ne peut faire apparaître une barre de défilement que si son parent a une taille définie en pixels. Comme l'objectif est de faire défiler horizontalement les exercices seulement si l'espace disponible n'est pas suffisant, il apparaît évident que figer la taille n'est pas acceptable.

La solution que j'ai retenue consiste à utiliser des tables pour effectuer la mise en page, contrairement à la préconisation de centaines d'internautes [**STACKOVERFLOW15**]. Pour que l'ensemble fonctionne comme attendu, le style de la table qui contient le conteneur "`<div>`" doit être défini exactement comme suit : "`table-layout:fixed;width:100%;`". Les deux consignes me paraissent contradictoires, mais cela fonctionne.

Pour synthétiser, le grand nombre d'imbrication d'éléments, les styles provenant de Bootstrap, et les styles qui s'activent en fonction de la résolution rendent extrêmement difficiles les interventions sur la mise en page. Des tâches *a priori* simples peuvent consommer des journées entières.

La mise en page est parfois immensément plus simple et intuitive qu'avec les CSS. Notamment celle de Microsoft en XAML pour Wpf et Silverlight. Une fois acquis les concepts des différents types de conteneurs (grille, empilement, décoration, répartition), les imbriquer et concevoir une interface qui se comporte parfaitement en cas de redimensionnement devient vraiment aisé. Ceci pour mettre en perspective la mise en page HTML et indiquer qu'elle est complexe, sinon "infernale".

2.2.2 PILOTAGE DU MICROMONDE TPELEC

2.2.2.1 État initial et contraintes

La simulation de circuit électrique *TPElec* a été développée par un autre groupe de travail de l'équipe MeTAH, et indépendamment de FORMID. Son fonctionnement interne se base sur un serveur qui contient les circuits électriques, et enregistre toutes les actions effectuées. Son acceptation dans FORMID, réalisée bien avant mon arrivée, a donné naissance à des services web dans la partie serveur de FORMID. En effet, le composant *TPElec* est conçu pour fonctionner en lien avec son propre serveur, pour charger et sauvegarder les circuits.

J'ai constaté que l'ensemble fonctionnait. Cependant, la gestion technique des dispositifs rompt la généricité à l'intérieur de FORMID, à cause de *TPElec*. Plus grave il apporte son lot de contraintes fortes en termes de fonctionnement pour FORMID.

En premier lieu, le nom du circuit à utiliser est fourni avec les autres paramètres, dans un attribut HTML nommé "`FlashVars`" (comprendre « variables pour le composant Adobe Flash ») qu'il n'est pas possible de modifier après le chargement initial de *TPElec*. Il n'est donc pas possible depuis FORMID de modifier le circuit dynamiquement pendant le travail avec *TPElec*.

Des fonctions importantes ont donc été mises de côté pour tous les dispositifs, à cause des contraintes que *TPElec* avaient engendrées par son implémentation. Tout d'abord côté élève :

- **Réessayer une étape** : pour réessayer il est nécessaire de placer l'apprenant dans la même situation qu'au début de l'étape, donc de définir l'état du circuit.
- **Affecter un état initial au début d'une étape** : cette fonction est nécessaire dans certains exercices pour changer l'état du circuit à chaque étape. Or, comme le circuit est passé en paramètre lors de l'initialisation du composant Flash, donc au début de l'exercice, il ne peut pas changer.
- **Passer une étape** : cela nécessite de pouvoir placer le composant dans l'état initial de l'étape qui va être commencée. En effet, dans une succession d'étapes, l'auteur peut décider de placer au début de l'étape 2 le circuit de la fin de l'étape 1.

Et puis côté auteur, la gestion est inextricable, et voici pourquoi. Lors de la création d'un nouveau scénario, l'auteur peut choisir de se baser sur un circuit existant. J'avais d'ailleurs à ce propos ajouté une liste déroulante présentant la liste des circuits disponibles sur le serveur. Quel que soit le circuit choisi, le composant *TPElec* est alors initialisé avec l'adresse et le nom de ce circuit sous la forme d'une adresse URL. À partir de ce moment-là, si l'auteur sauve le circuit, un service du serveur est invoqué avec ce nom de circuit. Vient alors l'heure du choix pour moi développeur :

- En décidant d'écraser le fichier du circuit à son emplacement d'origine, je perds le circuit d'origine. Cela est un problème car l'auteur l'a certainement modifié pour les besoins de son scénario. Et donc, si un autre scénario utilise ce circuit, il ne fonctionnera plus.
- En décidant de sauver le circuit à l'emplacement et sous le nom de mon choix, je me retrouve face au problème de pouvoir retrouver le fichier du circuit correspondant au scénario.

Pour compliquer encore le problème, le scénario lui-même subit plusieurs étapes. Tout d'abord, il est conçu, puis compilé, et enfin sauvegardé. J'ai effectué un travail côté serveur pour que ces différentes étapes se déroulent dans des dossiers différents. En effet, un seul répertoire contenait un ensemble de scénarios qu'il n'était pas possible d'identifier comme étant sauvegardé, compilé, ou en production (lié à un exercice publié). Une fois ce processus au point pour le scénario, il m'a paru logique que le circuit ne soit pas dans une base « commune » mais que son fichier soit enregistré à côté du scénario, quel que soit son emplacement. J'ai commencé des développements dans ce sens, jusqu'à une certaine réunion.

2.2.2.2 Point d'inflexion

En effet, grâce à une réunion de travail avec Anne Lejeune, j'ai eu un déclic et ma vision a basculé. En voici les explications.

Nous discutions du composant *TPElec* et de la spécificité des notions de topologie et d'état observé. Concrètement, le circuit représente les composants, les fils, et des valeurs comme la résistivité et la tension nominale, donc, en synthèse, l'état du micromonde. Alors que d'un autre côté l'état actuel contient les valeurs des tensions et des intensités, et d'autres données comme par exemple l'information de composant grillé.

Dans FORMID, l'auteur peut prendre des « photos » de l'état du dispositif externe (quel qu'il soit, *TPElec* ou autre) à un instant donné et les affecter à l'état initial d'une étape. L'objectif était bien de pouvoir placer le dispositif externe dans un état donné au début d'une étape.

En résumé, FORMID gère un circuit initial, qui ne sert que dans le cas de *TPElec*, et un état photographié, c'est-à-dire approximativement la liste des variables. Cette liste des variables suffit à initialiser une simulation, puisqu'une simulation est uniquement définie par la valeur d'une liste fixe de variables (Cf. explications de la partie 1.3.1).

Par contre, comme expliqué plus haut, l'état courant de *TPElec* est la résultante de la topologie du micromonde après son passage dans le moteur de simulation électrique. Donc enregistrer l'état courant ne pourra jamais permettre de placer *TPElec* dans un état voulu, ce serait comme essayer de changer seulement le résultat d'une addition sans changer les nombres additionnés.

Ma prise de conscience de cet état de fait, pendant la discussion que j'ai évoquée, m'a conduit à vouloir stocker la topologie, donc le circuit dans la « photo » de l'état initial, à la place des valeurs calculées. Comme je vais le présenter, cette volonté m'a m'amené à créer des évolutions qui remettront également en cause le besoin de stocker le fichier du circuit à côté de celui du scénario.

2.2.2.3 Évolution suivie

Il existe dans le composant *TPElec* une méthode pour lire l'état actuel du circuit sous forme XML. L'état est donc très facilement lu et stocké dans la « photo ».

Par contre il convenait de rester compatible avec la simulation *TPNum* (qui consiste à manipuler des billes à l'écran). Par conséquent, le clic sur le bouton pour faire la photo ne déclenche plus la lecture de la liste des variables, mais la lecture de la structure du micromonde. Donc dans le cas de *TPElec*, l'implémentation de cette méthode dans l'adaptateur de communication va lire le circuit en XML, et dans le cas de *TPNum* cette méthode sera implémentée de manière à renvoyer comme auparavant la liste des variables. Ce début de réalisation d'un patron de conception « Strategy » (Cf. partie 3.1.3.2 concernant les patrons de conception logicielle) a été poursuivi par la suite avec la transformation des adaptateurs de communication en contrôleurs AngularJS.

Certains obstacles techniques ont été levés, mais il subsiste la question de la méthode à adopter pour pouvoir réaffecter au composant *TPElec* le circuit stocké dans l'étape. Comme *TPElec* utilise le fichier de circuit dont l'URL lui est fournie lors de son instanciation, et que le circuit à affecter n'est pas dans un fichier mais sous forme XML déjà côté client, j'ai cherché à savoir si *TPElec* offrait cette possibilité.

J'ai donc décompilé le composant Flash *TPElec* avec un outil approprié [**JPEXS FREE FLASH DECOMPILER15**] afin de comprendre dans le code source son mode de fonctionnement. J'avais besoin de connaître les limites de son interface de communication avec FORMID. Je cherchais notamment s'il était capable de modifier une des "`FlashVars`" pendant son fonctionnement, pour modifier le nom du circuit, ou s'il était capable de recevoir la définition d'un circuit directement sous forme XML. Il n'avait ni l'un ni l'autre, je me suis donc proposé auprès

de mes responsables et du responsable du projet *TPElec*, Jean-Michel Adam**,** pour faire évoluer très légèrement ce composant. Comme cela n'a pas pu être concrétisé quand j'en avais besoin, j'ai pris les décisions suivantes, avec comme conséquence le fonctionnement représenté Figure 31 :

- J'enverrai le circuit XML vers le serveur pour le stocker dans un fichier temporaire, puis je fournirai le chemin de ce fichier temporaire comme circuit.
- Je réinitialiserai l'intégralité du composant *TPElec* lorsque j'ai besoin d'affecter un circuit. Pour cela je le supprime de la page et le recrée après une courte attente.
- Et comme conséquence directe, j'abandonne complètement la gestion des fichiers de circuit. Ils ne deviennent que des fichiers temporaires.

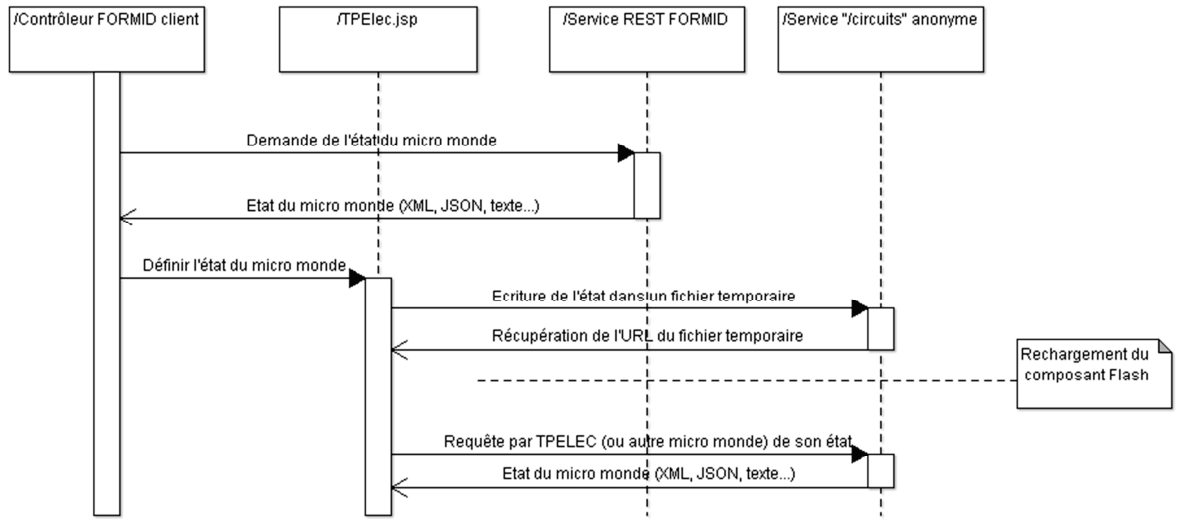


Figure 31 : la gestion des circuits de TPElec a été rendue générique.
Diagramme de séquence UML (le temps défile de haut en bas).

Cette solution est excellente pour plusieurs raisons.

- Tout d'abord, le champ « circuit d'un scénario », spécifique à *TPElec,* mais présent dans chaque scénario de saisie, peut être supprimé. L'ensemble devient dès lors générique.
- Ensuite, je résous tous les problèmes relatifs aux fichiers de circuits, puisque la définition du circuit est stockée à l'intérieur même du scénario (dans l'état initial d'une étape). Il n'y a plus de problème dû aux fonctions « ouvrir un circuit » et « sauvegarder un circuit » depuis le composant *TPElec* lui-même.
- Enfin, je suis maintenant capable de modifier l'état de *TPElec,* c'est-à-dire son circuit, à n'importe quel instant.

Avec ces grosses évolutions techniques, j'ai mis au point, corrigé et remis en service les éléments fonctionnels suivants :

- Définir l'état initial de chaque étape fonctionne. Idem pour l'état final.
- Le bouton « passer l'étape » trouve un nouveau souffle avec la possibilité précédente.
- Le bouton élève « recommencer l'étape » fonctionne dorénavant.

En conclusion, malgré une gestion alambiquée, le pilotage du micromonde *TPElec* s'effectue maintenant dans une bulle isolée du reste de FORMID. Toutes les contraintes de *TPElec* qui avaient bloqué certaines fonctionnalités de FORMID sont maintenant levées, et son fonctionnement en mode "boîte noire" (Cf. glossaire) est parfaitement générique.

2.2.3 MIGRATION DE LA BASE DE DONNÉES

La base de données H2 avait été choisie par mon prédécesseur parce qu'elle est nativement installée avec le serveur applicatif WildFly, et que cela lui faisait gagner du temps dans la réingénierie du logiciel. Cependant, des problèmes récurrents m'ont poussé à essayer de changer ce système de gestion de base de données.

La base H2 se présente sous la forme d'un fichier « JAR », c'est-à-dire une application Java. Ce n'est pas un service Windows ou un démon Linux qui tourne automatiquement en tâche de fond. Le fait de double-cliquer dessus démarre le moteur de base de données puis ouvre une page web sur son interface d'administration. Les inconvénients sont nombreux. L'arrêt de H2 ne fonctionne pas bien, il n'y a aucun journal d'erreurs, et pas de fichier de paramétrage. L'URL d'accès à notre base de données détermine le chemin physique dans le disque dur, mais si le fichier désigné n'existe pas, une nouvelle base de données est créée, sans message.

La procédure complète permettant ne serait-ce que d'ajouter un champ dans la base de données de production est impressionnante. Il est en effet nécessaire d'arrêter le serveur H2 sur le serveur Linux et en local, de sauvegarder la base locale, de rapatrier la base du serveur avant de pouvoir taper du SQL. Puis la procédure doit être recommencée dans l'autre sens.

J'ai donc eu assez vite le désir de faire migrer la base pour ne plus utiliser H2, à cause d'une certaine sensation de lourdeur. Mais la mise en application n'a pas pu être immédiate.

J'ai retenu les solutions MySQL et PostgreSQL [**POSTGRESQL15**], toutes deux légères, gratuites et avec une interface d'administration web. J'avais une préférence pour MySQL que je connaissais. En discutant avec mes responsables, il en est ressorti qu'Anne Lejeune maîtrisant bien PostgreSQL, le choix est donc devenu évident. En effet, je sais que les données sont extrêmement importantes, et en donner le contrôle aux responsables de FORMID me paraissait être une opportunité extrêmement intéressante.

Cette tâche n'était pas prioritaire, mais j'ai tenté de la faire progresser régulièrement. Cela m'a pris presque deux mois.

- **01/04/2015** Début de migration. Installation de PostgreSQL et phpPgAdmin sur Linux.
- **02/04/2015** Configuration d'Apache sur Linux pour autoriser une exception d'URL phpPgAdmin.
- **08/04/2015** Installation de PostgreSQL sur Windows, début de paramétrage de Maven, recherche de documentations, essai de lecture des tables H2 par un lien ODBC vers Microsoft Access.
- **13/04/2015** Ajout du module PostgreSQL dans Wildfly pour le déploiement de l'application.
- **26/05/2015** Travail sur la migration.
- **27/05/2015** Migration terminée avec utilisation de l'outil SquirrelSQL. Modification des fichiers de configuration de la couche de persistance de Java pour utiliser la nouvelle base PostgreSQL.

Le basculement pour FORMID de la base H2 vers PostgreSQL a été instantané, puisque FORMID utilise une couche de persistance de données générique, et que celle-ci peut fonctionner sur tout type de base de données. La difficulté à concevoir des requêtes génériques sans utiliser de SQL a donc été rentabilisée.

Tout cumulé, la migration de la base de données H2 vers PostgreSQL m'a pris un peu plus de 40 heures. La base de données du serveur de production FORMID est maintenant administrable directement via une page web.
J'ai beaucoup apprécié cette "victoire".

2.2.4 MANAGEMENT DES WEBSOCKETS

2.2.4.1 Présentation

Afin de bien comprendre les WebSockets, il convient de les replacer dans leur contexte. La Figure 32 montre la place des différentes méthodes de communication Web dans les couches réseau. Le protocole HTTP est le protocole historique, encore communément utilisé sur internet pour interroger les sites web.

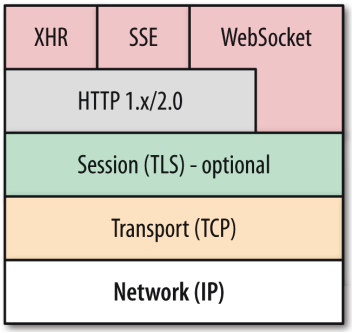


Figure 32: modèle des couches réseau des protocoles Web (HTTP), Ajax (XHR) et WebSocket.
La couche de base est Internet Protocol (IP). Source : http://realtimecloud.co.uk/talks/webrtc/bristech/#1 (Web, consulté en septembre 2015).

- Le protocole XHR signifie "XmlHttpRequest" [**MDN XMLHTTPREQUEST**]. Il s'agit de la capacité pour le navigateur d'interroger le serveur depuis un langage de script comme JavaScript.
- Le protocole SSE, qui signifie "Server-Sent Events" permet au navigateur de s'abonner à une page du serveur, qui devient alors capable de lui envoyer des notifications [**MDN SERVER-SENT EVENTS15**].
- Le protocole WebSocket [**WEBSOCKET15**] permet d'établir une communication bidirectionnelle entre un serveur web et les navigateurs qui y sont connectés.

Les WebSockets ont été implémentées dans FORMID par mon prédécesseur [**ECHTERBILLE14**]. La grande avancée des protocoles SSE et WebSocket par rapport au protocole HTTP est la possibilité offerte au serveur de prendre l'initiative de contacter le navigateur. La Figure 33 montre que les WebSockets combinent les capacités des méthodes XHR et SSE.

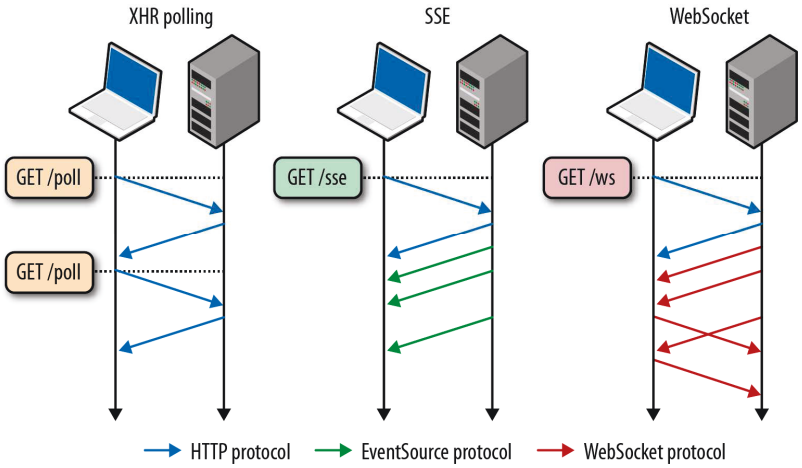


Figure 33 : protocoles de communication Web : XmlHttpRequest, Server-Sent Events, WebSocket.
Source: http://chimera.labs.oreilly.com/books/1230000000545/ch17.html#_request_and_response_streaming (Web), 09/2015

Offrir la possibilité au serveur de contacter les clients peut devenir très utile dans certains cas d'utilisation. Je vais maintenant exposer le problème.

2.2.4.2 Problématique

Lorsque j'ai pris mes fonctions au LIG, les WebSockets du serveur Linux ne fonctionnaient plus (Cf. Figure 34). Par conséquent c'est le module élève de FORMID qui ne fonctionnait pas.

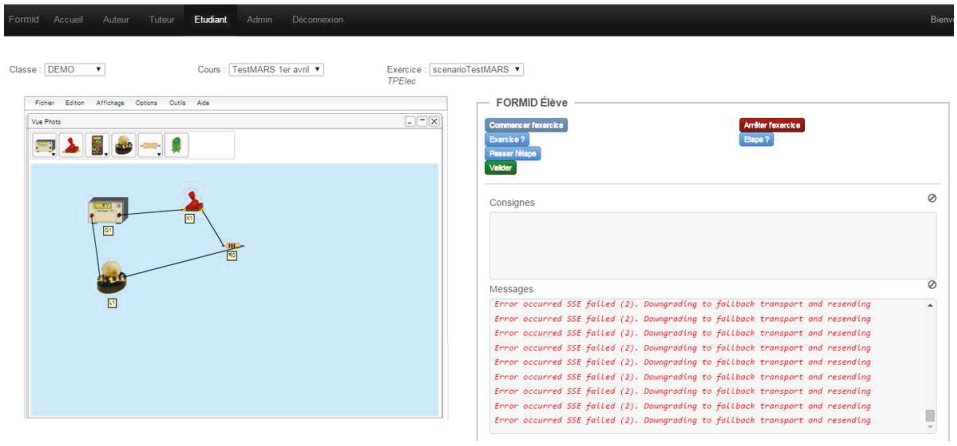


Figure 34 : illustration des erreurs de WebSockets dans FORMID.
Le mode dégradé ne semble pas fonctionner, donc l'erreur boucle.

Voici les étapes importantes de la gestion des WebSockets.

- **02 mars 2015** : À mon arrivée, les WebSockets sont en panne depuis une durée indéterminée.
- **03 avril 2015** : je constate que Apache 2.2.15 qui sert de proxy pour Wildfly, ne gère pas les tunnels WebSockets, et qu'il faut Apache 2.4.0. Mais en contactant Jean-Louis Mas, l'ingénieur support de l'équipe MISI (Moyens Informatiques et Systèmes d'Information), j'apprends que notre serveur Linux est une version CentOS6 "figée" qui est garantie sans évolution jusqu'en 2021.
- **08 avril 2015** : je trouve un code source créé par la communauté pour adjoindre un tunnel WebSocket à Apache 2.2.15. La communication est restaurée, le mode élève fonctionne.

Le module élève de FORMID a donc été en panne pendant plus d'un mois avant réparation. Comme les WebSockets ont eu fonctionné avec cette version de serveur, j'en ai déduit que mon prédécesseur avait trouvé une solution de contournement également, et que cette solution, quelle qu'elle fût, a été rendue inopérante par une action sur le serveur. Après ma remise en fonctionnement, nous avons tout de même continué à obtenir des erreurs de connexion WebSockets aléatoires. Mais comme je n'avais pas ce problème sur mon poste de développement – ou très rarement - même en interrogeant le serveur Linux, nous attribuions cela à la connexion Wi-Fi ou à des caprices de la machine.

Il a bien fallu nous rendre à l'évidence. Les problèmes n'étaient pas rares, et étaient même un peu trop présents pour être le fait du hasard. Sur le poste d'Anne Lejeune, le problème était récurrent, et ce, que la connexion réseau soit Wi-Fi ou filaire. De plus, un fait très gênant a été mis au jour : le mode dégradé ne fonctionne pas du tout. En effet, le choix de la librairie Atmosphere a été en partie guidé par le fait que si le navigateur ne gérait pas les WebSockets, la librairie savait utiliser la technique du "long-polling".

Les messages d'erreurs renvoyés par la librairie ne sont pas toujours les mêmes. Le message d'erreur SSE (Figure 34) semble indiquer que le SSE (Server Sent Event) est lui-même un mode dégradé des WebSockets, et ne fonctionne pas dans notre configuration.

Je suspecte le relais Apache comme étant en cause. Dans le fichier de configuration Apache (Cf. Figure 35), les adresses à faire transiter par le proxy sont déterminées par "`ws://`" ou "`http://`". Le protocole SSE [**W3C SERVER-SENT EVENTS15**] utilise des en-têtes particuliers du protocole HTTP. Or, il suffit qu'un seul en-tête requis par le protocole soit ajouté, modifié ou supprimé par le proxy pour que la connexion pose problème.

```
#NameVirtualHost [2001:660:5301:101::5]:80
#NameVirtualHost 129.88.101.5:80

<VirtualHost [2001:660:5301:101::5]:80 129.88.101.5:80>
    ServerName formid.imag.fr

    ProxyRequests Off

    <Proxy *>
    Order allow,deny
    Allow from all
    </Proxy>

ProxyPass /phpPgAdmin !

        ProxyPass        /ws/ ws://127.0.0.1:8080/ws/
        ProxyPassReverse /ws/ ws://127.0.0.1:8080/ws/

    ProxyPass           / http://127.0.0.1:8080/
    ProxyPassReverse    / http://127.0.0.1:8080/

    ErrorLog /usr/share/jboss/standalone/log/apacheLog
    CustomLog /usr/share/jboss/standalone/log/accessLog common
</VirtualHost>
```

Figure 35 : Fichier de configuration Apache pour la partie proxy.
Les lignes "ProxyPass" du fichier "/etc/httpd/conf.d/formid.conf" indiquent comment faire suivre les requêtes.
Les requêtes dont l'URL commence par "/ws/" seront relayées vers "ws://127.0.0.1:8080/ws/".
Les requêtes restantes seront redirigées vers "http://127.0.0.1:8080/".

Puis :

- **20 mai 2015** : la communication est tombée en panne. Je présume que le problème apparu avant mon arrivée s'est à nouveau produit. Je réalise donc les opérations de réinstallation du module Apache grâce à ma documentation, puis je redémarre les serveurs, et rédige une fiche de dépannage rapide. Le fonctionnement est restauré.

Le problème perdurait néanmoins. Mais il n'était pas bloquant car nous pouvions toujours trouver une voie de sortie : dans le pire des cas, au bout de cinq ou six essais de fermetures du navigateur, de redémarrage, de reconnexion à FORMID, en général la connexion WebSocket finissait par s'établir correctement.

2.2.4.3 Résolution

Le manque de fiabilité du fonctionnement des WebSockets nous est resté à l'esprit durant de longs mois. Il était difficile d'en comprendre la cause puisqu'elle était aléatoire et très peu présente sur le poste de développement.

- **12 août 2015 :** je développe un "mode élève déporté", c'est-à-dire la possibilité de profiter du scénario pédagogique sans que l'élève ne se connecte à FORMID depuis un navigateur. Il se connecte directement depuis un dispositif externe, comme une tablette tactile. Ce mode est développé de manière simple et générique, donc avec des services web et non des WebSockets.
- **01 septembre 2015** : je commence à soumettre l'idée de l'élimination des WebSockets qui posent beaucoup de problèmes. La communication n'est pas fiable, car de manière aléatoire elle refuse de s'initialiser.

- **02 septembre 2015** : j'analyse l'usage des WebSockets dans FORMID et je cherche à savoir si ma méthode de communication par service web développée pour le prototype du mode élève déporté pourrait être adaptée.

Avec la perspective d'une installation prévue fin septembre dans une école, et la répétition des problèmes rencontrés, conserver ce point faible devenait de plus en plus insupportable.

J'ai cherché à éliminer les sources potentielles de problème les unes après les autres. Le Tableau 4 liste les sources que j'ai estimées possibles. Le problème est qu'elles sont nombreuses et que beaucoup semblent avoir un impact aggravant mais non déterminant.

Tableau 4 : liste des causes possibles identifiées du problème récurrent des WebSockets.

Cause possible	Analyse et test
L'antivirus « Symantec endpoint protection » bloque la connexion de manière aléatoire.	J'avais réduit le rayon d'action de l'antivirus sur mon poste pour un autre problème. Nous avons comparé les réglages, puis fait des tests sur le poste d'Anne Lejeune avec l'anti-virus désactivé. Cela s'est mis à fonctionner bien mieux mais nous avions encore des problèmes dans 20% des essais.
Le proxy Apache avec le tunnel WebSockets ne fonctionne pas bien	C'est certainement un facteur aggravant d'ajouter un intermédiaire dans une connexion TCP/IP, mais ce n'est pas la seule cause car j'ai obtenu l'erreur également en local sur mon poste.
La connexion Wi-Fi est en cause	Nous avons eu le sentiment qu'effectivement, en Wi-Fi le module élève fonctionnait très rarement. Mais ça n'est pas le seul facteur car en filaire nous avons eu de nombreux problèmes.
La librairie WebSockets "Atmosphère" contient des anomalies	La mise à jour a été catastrophique et m'a ajouté des erreurs, donc j'ai gardé la version d'origine.
Mauvaise implémentation dans le code qui ne nettoie pas bien les connexions après usage.	Si cela était le cas, le fait de redémarrer l'environnement de développement suffirait à refaire tout fonctionner, mais cela n'a pas été le cas.
Le navigateur n'est pas à jour ou ne gère pas bien les WebSockets.	Nous avons comparé nos versions de Chrome qui étaient identiques avec Anne Lejeune. Et le problème apparaît aussi bien sur Chrome que sur FireFox.
Problème de topologie réseau avec les VLAN	Les serveurs virtuels sont isolés dans un VLAN, et utiliser le Wi-Fi fait passer par un autre VLAN également… pour tous ces mécanismes de connexion, les routeurs utilisés peuvent avoir un impact fort. Mais cela ne peut pas être l'unique cause car j'ai eu des problèmes aussi en local sur mon poste de développement.

Je ne suis pas parvenu à déterminer de manière certaine quel facteur pouvait être déterminant et quel autre pouvait être négligé. Il existe forcément au moins un problème d'implémentation soit dans le code de FORMID soit dans la librairie Atmosphère, mais cela ne peut pas être la seule cause car la grande majorité du temps mon poste de développement fonctionnait très bien.

La recherche d'une solution m'a poussé à analyser en quoi la spécificité des WebSockets, à savoir la capacité du serveur à initier la communication vers le client, était utilisée. Il s'est révélé que cette propriété n'était pas utile à FORMID. Il a finalement été décidé de le supprimer à la source en s'affranchissant des WebSockets.

- **15 septembre 2015** : à l'ordre du jour, la 7ème priorité concerne la recherche d'une solution pour se passer des WebSockets.
- **18 septembre 2015** : le problème de communication devient très présent sur mon poste de développement, alors qu'il était jusque-là généralement apparu en production. Je cherche à mettre à jour la librairie "Atmosphère" qui gère cet aspect. C'est catastrophique, de nouvelles erreurs apparaissent. Notamment, la librairie Atmosphère côté client contacte des milliers de fois par seconde le serveur pour maintenir la connexion active. Le problème est connu sur les forums. Le concepteur indique comment résoudre le problème, mais cela ne fonctionne pas. Je reviens à la version historique.
- **21 septembre 2015** : La décision est prise, je commence une tentative de migration en service web.
- **22 septembre 2015** : FORMID fonctionne maintenant sans WebSockets.

Ce choix a des impacts. Tout d'abord, il n'est plus possible pour le serveur de contacter le client FORMID. Si cela devait être fait dans l'immédiat, il faudrait que le client interroge régulièrement le serveur pour savoir s'il a un message. Mais ce besoin n'est pas présent et non prévu dans l'immédiat.

J'avais le choix entre laisser le code des WebSockets en sommeil pour éventuellement un jour le rependre, ou le supprimer proprement mais obliger à tout redévelopper en cas de besoin. Dans l'immédiat, j'ai opté pour la première option.

2.2.5 LIAISON À UNE SIMULATION SUR TABLETTE TACTILE

Suivre la ligne directrice qui consiste à rendre FORMID capable d'utiliser la simulation *SimuBûchettes* sur tablette tactile a mené à de nombreuses améliorations successives. Je vais expliquer ces différentes étapes dans l'ordre où elles se sont déroulées, pour conserver une cohérence et faciliter la compréhension globale de la démarche.

La simulation *SimuBûchettes*, sur tablette tactile, et sous système d'exploitation Android, a été développée par Myriam Clouet pendant son stage de M1 mi-2015, sous la direction de Pierre Tchounikine. Cette simulation vise à apprendre la numération aux élèves de primaire, par la manipulation de bûchettes et de fagots virtuels.

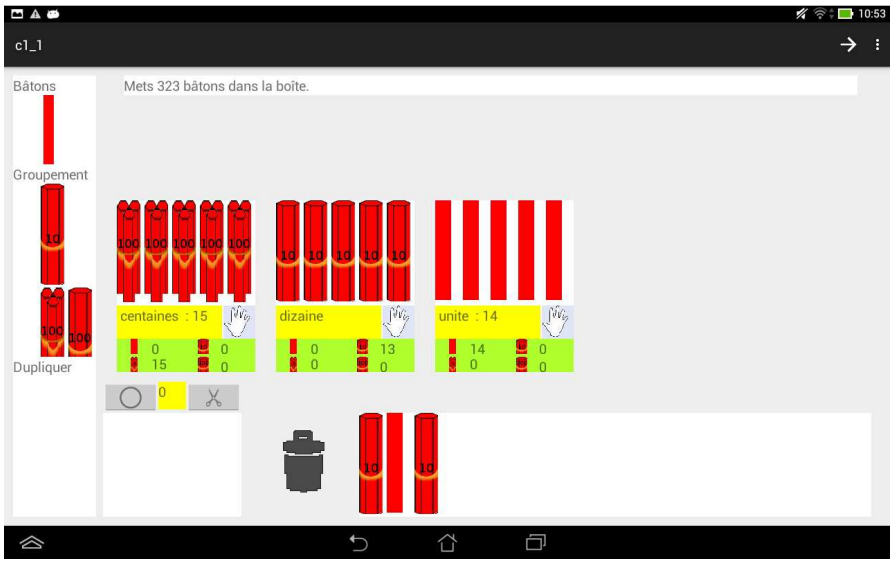


Figure 36 : capture d'écran de la simulation SimuBûchettes sur Android.

Sur la Figure 36, l'élève peut déplacer des bûchettes, les regrouper en fagots grâce à un élastique et les séparer à nouveau avec les ciseaux. Des consignes lui sont affichées en haut de l'écran. Il doit donc placer les quantités demandées dans différentes boîtes aux rôles paramétrables (sur l'exemple ce sont les centaines, dizaines et unités). Des expérimentations semble-t-il concluantes ont été réalisées en classe juste avant l'été 2015.

2.2.5.1 Projet de liaison avec FORMID

Comme ce projet est complètement indépendant de FORMID, nous avons eu connaissance de son existence et de son mode de fonctionnement après les premières expérimentations. Avec Anne Lejeune et Viviane Guéraud, nous avons discuté de l'intérêt que cela pourrait représenter de se lier à cette simulation pour la rendre utilisable dans les scénarios pédagogiques de FORMID.

2.2.5.2 Problématique de la communication

Après réflexion, il me paraissait illogique de chercher une solution technique pour que FORMID puisse interroger un logiciel sur tablette Android plusieurs fois par seconde, puisque cela revenait conceptuellement à transformer la tablette en serveur. Ce qui n'est vraisemblablement pas la bonne philosophie. En effet, tout d'abord, la tablette possède une source d'énergie autonome, ce qui induit sa mise en veille rapide et fréquente, et ensuite, il faudrait que la tablette diffuse son nom ou dispose d'une IP pré-paramétrée pour qu'il soit possible de la joindre. Enfin, il faudrait à la fois qu'il soit possible d'initier une connexion vers le réseau virtuel Wi-Fi, et que le logiciel sur tablette puisse se mettre en écoute. Toutes ces raisons m'ont convaincu de trouver une solution bien plus adaptée.

2.2.5.3 Projet d'évolution de FORMID

J'ai donc commencé par rédiger le document de 15 pages « *Dialogue avec les dispositifs externes* » tenant lieu à la fois de documentation et de spécification. J'y décris le mode de communication actuel, que j'appelle « FORMID actif » (c'est FORMID qui interroge le dispositif externe), et un projet de développement dans lequel FORMID pourrait être également passif (attendre qu'un dispositif externe le contacte). L'annexe 5 montre les diagrammes de séquence UML illustrant ce propos.

Je me suis proposé pour réaliser la partie communication de *SimuBûchettes*, qui se développe sur Android Studio, afin de maximiser les chances que FORMID puisse s'ouvrir de nouvelles perspectives. Les accords stratégiques et politiques prenant parfois un peu de temps, je me suis trouvé en situation d'attente.

2.2.5.4 La simulation *LettersGame* comme preuve de concept

Le développement côté FORMID étant important, j'ai eu l'idée de créer une nouvelle micro-simulation qui me servirait de bac à sable, c'est-à-dire de support pour expérimenter les différentes évolutions, dont fait partie le « mode élève déporté » que je vais expliquer.

J'ai arrêté mon choix sur une simulation qui contiendrait un ou plusieurs plateaux de jeu, avec des cases de couleur sur lesquelles il serait possible de déplacer des lettres. La première version fonctionnelle est née en deux jours. C'était un investissement intéressant puisque j'ai immédiatement pu faire évoluer FORMID comme imaginé (Annexe 6).

La simulation *LettersGame* (Cf. Figure 37) a été développée avec deux modes de communication. En mode interne, elle s'affiche au sein de FORMID et dialogue directement en JavaScript. En mode externe, elle s'affiche dans une page web séparée, et contacte le serveur FORMID pour lui notifier ses changements d'état, et lire ses messages. En effet, FORMID a besoin de définir l'état du dispositif externe pendant les exercices, et pour ce faire il laisse un message sur le serveur, qui sera transmis au dispositif externe dès qu'il se manifeste.

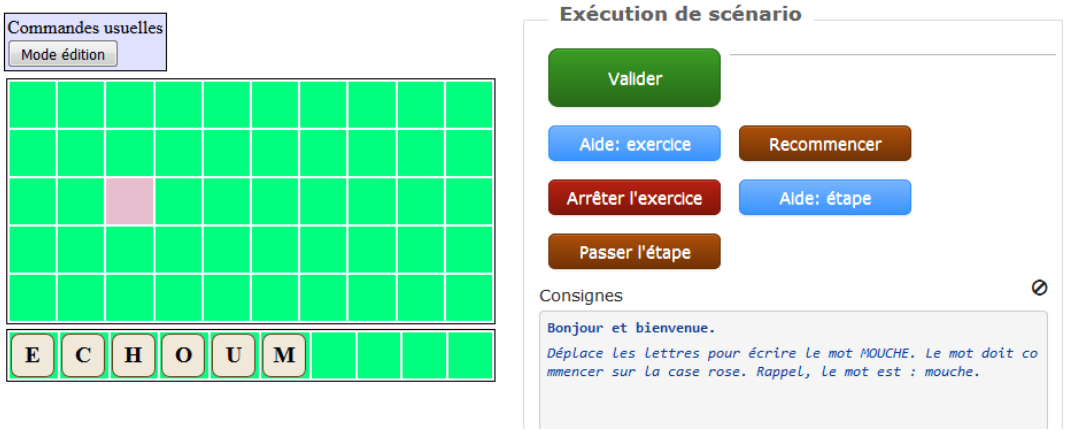


Figure 37 : exemple d'exercice avec la simulation *LettersGame*.

Ce nouveau mode de communication a introduit de nouvelles problématiques. En effet, compte tenu du fait que plusieurs instances de *LettersGame* peuvent coexister, se posent les questions de savoir comment les identifier et comment choisir celle à utiliser pour une session de travail FORMID. Le principe retenu consiste à considérer que chaque instance a un identifiant unique (alphanumérique), puis à afficher la liste des dispositifs externes accompagnés de leur identifiant et de leurs variables, pour aider à choisir le bon. La Figure 38 montre comment le choix du dispositif externe se présente à l'élève ou à l'auteur. Lorsque l'identifiant est le nom de l'élève, le choix en est grandement facilité.

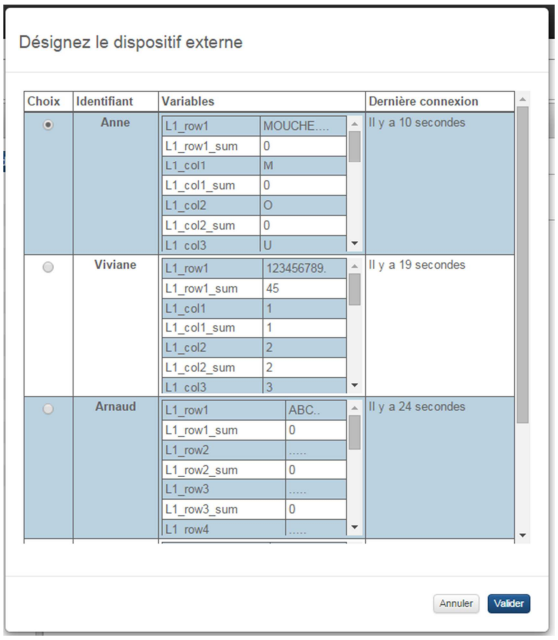


Figure 38 : écran de choix d'une instance de dispositif externe pour un utilisateur de FORMID.

La simulation *LettersGame* fonctionne donc correctement en mode externe. Il est possible d'ouvrir autant de pages que nécessaire, et d'utiliser l'une des instances depuis *FORMID* pour une session de travail.

2.2.5.5 Le mode élève déporté

Suite à ces réalisations fonctionnelles, nous nous sommes interrogés, mes responsables et moi-même, sur l'utilisabilité future de FORMID conjointement à une tablette tactile. En effet, la manipulation des bûchettes doit être réalisée sur la tablette tactile, mais les consignes de l'exercice, les boutons d'action et les messages de retour s'affichent dans l'application web FORMID.

J'ai proposé qu'il soit possible de déporter les commandes principales du mode élève à l'intérieur du dispositif externe. La proposition était étayée d'un document de projet technique. Ainsi, dans le cas de la tablette tactile, il ne serait plus nécessaire d'ouvrir FORMID sur un ordinateur à côté pour profiter du scénario pédagogique. Par contre, comme le mode élève de FORMID utilise exclusivement les WebSockets, il a fallu, pour permettre le fonctionnement décrit en Annexe 7, offrir la possibilité d'exécuter le mode élève grâce à des services web, c'est-à-dire des requêtes HTTP communes. C'est cette base technique qui a facilité par la suite un fonctionnement de FORMID sans les WebSockets.

Dans un premier temps, c'est une preuve de concept qui a été réalisée puisque tous les problèmes n'ont pas été résolus. Le plus important est la problématique du lien profond avec les données de FORMID. En effet, un élève dans FORMID doit être identifié formellement, ainsi que la séance et l'exercice à exécuter. Ce sont les données minimales nécessaires pour que les traces de l'élève s'enregistrent, et soient notamment visualisables par le tuteur.

Ma problématique était d'arriver à proposer des fonctionnalités avancées d'intégration dans un dispositif externe, à savoir identifier l'élève, choisir un groupe, une séance, et un exercice, sans que le dispositif externe soit envahi de connaissances fonctionnelles inhérentes à FORMID. Il me fallait trouver une solution peu intrusive pour qu'elle soit facilement acceptée et praticable dans de nouveaux dispositifs externes.

J'ai finalement trouvé la solution suivante, que j'ai immédiatement conservée. La contrainte est de développer dans le dispositif externe la possibilité d'afficher une liste à choix multiples pour l'utilisateur. La sémantique des données est connue uniquement de FORMID. Le fonctionnement est ensuite simple. Lorsque l'élève appuie sur

le bouton « *Commencer l'exercice* » dans le dispositif externe, FORMID évalue s'il a suffisamment de données pour le faire. Sinon il envoie selon le cas, une liste d'utilisateurs, de groupes, de séances, ou d'exercices pour que l'utilisateur choisisse. L'intérêt principal est que FORMID s'adapte. Par exemple si l'utilisateur ne peut travailler que dans une seule séance, le choix est automatique, de même s'il n'y a qu'un seul exercice dans la séance.

Je n'ai pas identifié d'inconvénients, par contre les avantages me paraissent vraiment évidents :

- Seules les données manquantes à FORMID sont demandées à l'utilisateur.
- Si des mécanismes de liaison sont mis en place, comme l'utilisation du même nom d'élève dans FORMID et dans le dispositif externe, alors des étapes de choix disparaîtront naturellement.
- Le fonctionnement complet de FORMID peut évoluer sans qu'on ait à retoucher le code du dispositif externe. Par exemple il pourrait être décidé de demander à l'utilisateur s'il souhaite être en mode normal ou en mode entraînement. Ajouter cette étape serait vraiment facile.
- Le mécanisme a facilement évolué pour que l'utilisateur puisse saisir son mot de passe. Dans ce cas le dispositif externe doit aussi demander un choix à l'utilisateur non issu d'une liste fermée.
- L'intrusion de FORMID dans le dispositif externe est minimaliste.

Le mode élève déporté peut donc être implémenté de manière fonctionnelle.

2.2.5.6 Amélioration de *SimuBûchettes*

Avec le temps la situation a évolué. Voici les dates notables :

- **27 juillet 2015** : développement de *LettersGame* pour faire évoluer FORMID en prévision de *SimuBûchettes*.
- **16 septembre 2015** : le code de *SimuBûchettes* est mis à ma disposition.
- **17 septembre 2015** : La simulation *SimuBûchettes* est maintenant utilisable dans FORMID.

Grâce à l'ensemble des évolutions précédemment détaillées, la liaison de *SimuBûchettes* à FORMID a presque été une formalité. Il a tout d'abord fallu intégrer dans le code *SimuBûchettes* une routine de notification de FORMID lorsque l'état de la simulation change. Ceci en langage Java dans Android Studio, et de manière à ne jamais bloquer l'interface utilisateur. Ensuite, il a fallu écrire l'adaptateur de communication côté FORMID.

Toutes les interactions décrites dans le diagramme de séquence UML en Annexe 8 sont mises en pratique dans le cas de *SimuBûchettes*.

2.2.5.7 Synthèse

La création de la simulation *LettersGame* pour permettre de faire évoluer FORMID a été très bénéfique. Même si en durée absolue il a fallu écrire une simulation en plus, en termes d'échéance j'ai pu commencer les évolutions techniques un mois et demi avant la mise à disposition du code source de *SimuBûchettes*.

La possibilité offerte par le mode élève déporté dans le dispositif externe est donc fonctionnelle dans *LettersGame*, mais elle n'a pas été implémentée dans *SimuBûchettes*. En effet, suite à une réunion d'équipe MeTAH, mesdames Guéraud et Lejeune ont discuté avec moi de l'avenir de cette voie. Il apparaît que cela contredit le message public véhiculé par FORMID, à savoir que cette application est capable de s'interfacer avec des dispositifs externes quelconques préexistants, sans les modifier. Et en effet, il est nécessaire de faire évoluer le dispositif externe sur plusieurs aspects.

Le mode élève déporté semble donc pour le moins mis en hibernation. Je ne considère pas pour autant qu'il s'agisse d'un travail réalisé à perte. D'une part la preuve est faite que FORMID peut offrir la puissance de son scénario pédagogique, sans pour autant imposer l'utilisation de son application web. Et d'autre part je suis persuadé qu'il est fort probable que le besoin émerge. Et surtout, le mode élève déporté m'a contraint à mettre en place tout un mode de communication par services web. Et ceci s'est révélé comme un travail préparatoire valorisé dans la tâche consistant à s'affranchir des WebSockets.

Dans le même temps, FORMID est maintenant parfaitement en mesure d'observer la simulation *SimuBûchettes,* et donc de construire un scénario avec ce nouveau dispositif externe.

En bonus, la simulation *LettersGame*, créée au départ pour les besoins que j'ai mentionnés, s'est révélée très intéressante en tant que telle. Elle permet en effet de déplacer des lettres et des chiffres sur des plateaux paramétrables, et autorise donc des utilisations aussi diverses qu'écrire un mot, résoudre un carré magique, poser une addition ou remplir une grille de mots croisés.

2.2.6 CONCLUSION

Les évolutions techniques réalisées ont été nombreuses, et pour certaines plutôt complexes. L'avancée par étapes a néanmoins permis de trouver des solutions à tous les problèmes, et de faire évoluer efficacement le logiciel. L'expérience a montré que certaines tâches a priori marginales en ont facilité d'autres par la suite. Notamment le fonctionnement sans WebSockets a été rendu possible par le développement préalable du mode écoute de FORMID, puis du mode élève déporté.

2.3 CONCLUSION DU CHAPITRE

Les réalisations les plus importantes viennent d'être détaillées d'un point de vue fonctionnel puis technique. Nous retiendrons dans les avancées majeures l'amélioration du mode auteur, et l'ouverture de FORMID à toujours plus de dispositifs externes (catalogués dans l'Annexe 13). La fiabilité de FORMID a été grandement améliorée par la suppression des WebSockets, et l'élimination de H2 a permis de s'affranchir des lourdeurs d'administration de la base de données. Je vais maintenant faire un état des lieux final de FORMID, et en profiter pour évoquer tous les concepts d'ingénierie logicielle rencontrés pendant la phase de réalisation.

3 ETAT DES LIEUX

Je voudrais dans cette partie passer en revue les éléments constituant la situation telle que je la laisse, que ce soit le modèle de la base de données ou les capacités actuelles de gestion des dispositifs externes. Je voudrais également présenter les concepts théoriques dont des implémentations ont été rencontrées ou utilisées dans FORMID. Et enfin, j'évoquerai la gestion de l'information.

3.1 CONCEPTS MANIPULÉS

La maîtrise technique en général passe par la compréhension théorique de l'architecture logicielle et de patrons de conception. L'utilisation de cadriciels bien conçus peut confronter le développeur à de nombreux concepts simultanément. C'est le cas avec AngularJS, c'est pourquoi je vais en faire une présentation, avant de parler des éléments d'ingénierie logicielle d'une manière plus théorique.

3.1.1 ANGULARJS

AngularJS est un cadriciel JavaScript développé par Google. Il apporte de grandes fonctionnalités et il est fortement structurant. Son apprentissage nous confronte à des notions complexes qui se présentent de manière inopinée pendant le développement. Ces surprises sont effrayantes parce qu'elles donnent parfois la sensation d'être complètement bloqué, ce qui donne après coup lieu à des traits d'humour, comme illustré Figure 39. La maîtrise complète de ce cadriciel requiert énormément d'investissement.

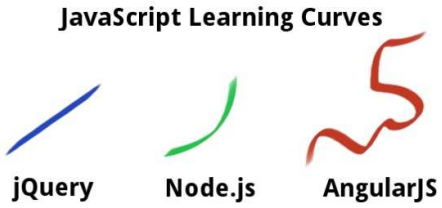


Figure 39: vue d'artiste de la courbe d'apprentissage d'AngularJS représentant sa difficulté masquée.
Source: http://nathanleclaire.com/blog/2014/01/04/5-smooth-angularjs-application-tips/

La prise en main peut être facilitée pour quelqu'un ayant déjà appréhendé les mêmes concepts ailleurs. Le site TodoMVC [**TODOMVC15**] expérimente la réalisation d'un même programme « liste des choses à faire » avec de nombreux cadriciels différents. Les nouveaux cadriciels étant conçus à partir des bonnes idées des anciens et des bonnes pratiques d'architecture logicielle, cela mène tout logiquement à de nombreux points communs. Le composant « Ember.js » [**EMBER.JS15**] par exemple gère également la liaison de données, le routage de pages, et les modèles de blocs HTML.

3.1.1.1 Les principaux concepts d'AngularJS

Je vais commencer par expliquer les caractéristiques du cadriciel, puis je reviendrai sur le sujet dans un exemple articulant l'ensemble.

Organisation du site

Tout d'abord, une contrainte importante d'AngularJS est qu'une application est destinée à fonctionner dans une seule page web. La clé de cette possibilité est le chargement dynamique de parties HTML nommées vues partielles. Par exemple, une boîte à onglets peut définir une vue partielle pour chaque onglet. Le cadriciel téléchargera et intégrera automatiquement, au besoin, le contenu HTML provenant de chaque fichier spécifié.

Architecture Modèle-Vue-Contrôleur (MVC)

Une application AngularJS déclare la « Vue », donc le HTML, d'un côté, et le « Contrôleur », c'est-à-dire le code JavaScript de l'autre. Le « Modèle » est le serveur fournissant les données. Le liant entre le contrôleur et la vue est assuré par le cadriciel AngularJS, grâce à la notion de directive.

Directives

Les directives sont au centre du fonctionnement. Une directive est une information HTML qui déclenche un comportement. L'information peut être de différents ordres : un élément ou un attribut HTML, ou même une classe CSS. L'attribut `ng-controller` indique notamment quel contrôleur AngularJS doit être instancié pour gérer un bloc d'affichage. Il est possible de créer des directives personnalisées qui ajoutent des blocs HTML avec un comportement programmé, comme la directive `i18n` de FORMID. Par exemple avec ce morceau HTML :

```
<span i18n='admin.student.firstname'></span>
```

La directive `i18n` aura comme comportement d'aller rechercher le libellé dans la langue en cours, par exemple "prénom", et de le mettre à l'intérieur du `span` pour qu'il puisse s'afficher.

Liaison de données

Il n'est pas question avec AngularJS d'accéder directement à une zone de saisie pour lire ou écrire sa valeur, comme nous pourrions le faire avec jQuery. Chaque zone de saisie doit déclarer un « modèle », qui désigne une variable dans le contrôleur courant. La liaison de données permet l'affichage immédiat d'une variable lorsqu'elle est modifiée dans le contrôleur, et la récupération automatique de la valeur d'une zone de saisie.

Injection de dépendances

La conception est très modulaire. Une application se compose principalement de contrôleurs et de services, dont certains sont fournis par le cadriciel. Un service peut être consommé dans un autre service ou dans un contrôleur. Le développeur ne gère absolument pas les instances de ces composants. Quand le cadriciel instancie un contrôleur par exemple, il passe en paramètre les instances des éléments déclarés par le développeur comme étant requis. Le cadriciel se charge de gérer les instances en amont pour pouvoir les fournir. C'est l'injection de dépendances.

Les « promesses »

Bien qu'incontournables, les promesses (*Promise* dans le code) ne sont pas propres à AngularJS puisqu'elles existent sous des formes semblables dans JavaScript et dans jQuery. Elles seront donc présentées de manière générique.

3.1.1.2 Exemple complet

Nous allons maintenant mettre en œuvre les concepts précédemment décrits. La Figure 40 montre un exemple minimaliste d'une page web composée de HTML et de JavaScript. Les attributs `ng-app` et `ng-controller` sont des directives interprétées par AngularJS, et qui servent à instancier (créer) le contrôleur `studentController` et le lier à la vue. Le paramètre `$scope` du constructeur, qui donne accès au contexte du contrôleur, est fourni par le moteur d'injection de dépendances.

Vue HTML "index.html" :

```
<div ng-app = "mainApp" ng-controller = "studentController">
   Enter first name: <input type = "text" ng-model = "student.firstName"><br><br>
   Enter last name: <input type = "text" ng-model = "student.lastName"><br>
   <br>

   You are entering: {{student.fullName()}}
</div>
```

Contrôleur JavaScript "app.js" :

```
<script>
   var mainApp = angular.module("mainApp", []);

   mainApp.controller('studentController', function($scope) {
      $scope.student = {
         firstName: "Mahesh",
         lastName: "Parashar",

         fullName: function() {
            var studentObject;
            studentObject = $scope.student;
            return studentObject.firstName + " " + studentObject.lastName;
         }
      };
   });
</script>
```

Résultat à l'écran :

Figure 40 : exemple d'application AngularJS avec mise à jour du nom complet pendant la frappe.
La vue (HTML) utilise des directives pour se lier au contrôleur (JavaScript).
Source : http://www.tutorialspoint.com/angularjs/angularjs_controllers.htm (consulté en octobre 2015).

La liaison de données affecte « Mahesh » et « Parashar » aux zones de saisie à l'ouverture, puis mettra à jour automatiquement la variable « `$scope.student` » lors de la modification par l'utilisateur. Cette mise à jour aura pour effet immédiat l'évaluation de la méthode `fullName()` dont le résultat s'affichera dans la vue à la place de l'expression entre double accolade.

Le cadriciel AngularJS est riche, car en plus de comprendre d'autres notions, il permet une utilisation très fine de celles qui viennent d'être présentées. Une directive notamment peut être extrêmement complexe à écrire, et des directives peuvent en contenir d'autres.

3.1.2 LA COMMUNICATION ASYNCHRONE

L'objectif d'un programme web est de ne jamais bloquer l'interface visuelle afin de fournir une bonne expérience utilisateur. En effet, si une donnée est chargée de manière synchrone sur le serveur, la vue ne pourra pas réagir aux actions de l'utilisateur, et semblera être figée tant que la réponse n'aura pas été obtenue.

C'est pourquoi l'ensemble des développements devraient de nos jours être réalisés en mode asynchrone. L'inconvénient est qu'il est bien plus complexe d'écrire une suite d'instructions lorsque le résultat de la première n'est pas immédiatement disponible pour la seconde.

Les différents langages de développement ont évolué en conséquence et proposent des solutions permettant de simplifier l'écriture du code. Pour aller dans l'extrême, le Framework 4.5 de Microsoft permet grâce aux instructions `async` et `await` d'écrire une suite d'actions asynchrones comme s'il s'agissait d'un programme synchrone [**MSDN ASYNC AWAIT15**].

En JavaScript, l'usage historique comprend deux méthodes :

- La première est de spécifier par « abonnement » quelle fonction il est nécessaire d'invoquer à chaque fois qu'un certain traitement sera terminé. L'inconvénient principal est la nécessité au moment de la réception de la réponse d'identifier quelle était la requête, pour savoir que faire du résultat. L'enchaînement d'appels se révèle très complexe, tout comme la capture d'exceptions.
- La seconde façon est d'appeler une fonction, en lui disant quelle autre fonction invoquer pour répondre. Il s'agit de la méthode par *Callback* (« recontacter » en français). Bien que plus fonctionnelle, cette méthode pose tout de même son lot de problèmes : comme le prototypage (la liste des paramètres) de la fonction de rappel est libre, il ne peut pas exister de cohérence entre les différents usages, et le développeur ne peut donc pas savoir quel prototypage est attendu. C'est une source d'erreurs importante.

Les promesses apportent une solution différente et tout à fait intéressante. Elles sont très récentes dans le langage JavaScript, mais supportées par plusieurs navigateurs comme Google Chrome et Mozilla FireFox (elles ne sont par contre pas supportées par Microsoft Internet Explorer, version 11 comprise).

L'idée générale d'une promesse est d'invoquer une fonction en écrivant le « ensuite », c'est-à-dire en spécifiant qu'il faudra exécuter telle suite d'instructions quand le traitement sera terminé. La Figure 41 montre un exemple de syntaxe d'une promesse. Elle fournit en paramètre de son corps deux fonctions à appeler, « `resolve()` » si tout s'est bien passé, et « `reject()` » en cas d'erreur. Lorsque la première est invoquée, alors le code présent dans le « `then` » de l'appelant est exécuté. Il est également possible de déclarer une deuxième fonction dans le « `then` », qui elle sera exécutée en cas d'échec.

```
// Déclaration d'une fonction qui renvoie une promesse
// Cette fonction attend pendant un délai spécifié par 'duration', en millisecondes
function timeout(duration) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() { resolve(); }, duration);
    });
}

// Utilisation de la promesse retournée par la fonction pour afficher un message
// lorsque 5 secondes se seront écoulées
timeout(5000).then(function() {
    console.log("Le délai est écoulé");
})
```

Figure 41 : utilisation d'une promesse en JavaScript pour exécuter du code après un délai choisi.

Une promesse est facilement compréhensible, quel que soit le programme dans lequel elle est utilisée. Elle procure plusieurs avantages : la gestion d'erreur est intégrée, et elle est également facilement chaînable. Dans l'exemple précédent, il serait facile d'écrire à la place du « `console.log` » l'appel à une autre fonction asynchrone. Les promesses sont très flexibles. Elles peuvent contenir du code synchrone, ou même être déjà terminées lorsqu'elles sont renvoyées, sans que cela pose de problème. Donc utiliser des promesses même en mode synchrone permet de préparer les évolutions futures.

La syntaxe diffère quelque peu entre JavaScript, jQuery et AngularJS, mais le mécanisme reste le même. Les promesses sont très largement utilisées dans FORMID, déjà parce qu'elles font partie intégrante des cadriciels, mais également parce que je les ai utilisées systématiquement pour gérer les dispositifs externes.

Initialement, la communication de FORMID avec le dispositif externe était synchrone, il n'y avait pas de promesses, de notion de dispositif « prêt » et de dispositifs multiples. Le diagramme déjà présenté en Annexe 8 montre le mécanisme final tel que je l'ai mis en place pour gérer les dispositifs externes de manière asynchrone grâce à des promesses.

En tout état de cause, les promesses sont au cœur de ma nouvelle gestion des dispositifs externes, et je n'ai pas constaté le moindre inconvénient quant à leur usage intensif ou à l'empilement de leurs appels.

3.1.3 GÉNIE LOGICIEL

Le génie logiciel est une discipline étudiant notamment les problèmes récurrents, et les bonnes pratiques admises pour les résoudre. Utiliser un patron de conception (« *design pattern* ») permet de s'assurer que la méthode utilisée est éprouvée, et facilite également la compréhension par tous les autres acteurs qui connaissent le concept. Précisons que le patron doit pour cela être adapté à la situation, et bien réalisé. En outre, l'architecture logicielle comprend l'organisation des différents composants ou des différentes couches d'un programme.

3.1.3.1 Modèle MVC et Modèle en couches

L'architecture MVC (« *Model-View-Controller* »), inventée en 1978 par Trygve Reenskaug [**REENSKAUG07**], a un intérêt qui n'est plus à démontrer puisqu'elle est utilisée dans une majorité de projets de développement. L'idée de base très simple consiste à séparer les données (Modèle), la présentation (Vue) et les traitements (Contrôleur). Nous noterons que cette volonté est toujours d'actualité puisqu'elle se retrouve dans d'autres modèles récents comme MVVM (« *Model-View-ViewModel* ») créé en 2004 par Microsoft. Dans MVVM s'ajoute la volonté de minimiser le couplage entre le contrôleur de la vue (« *ViewModel* ») et la vue (« *View* »), avec comme objectif la capacité de remplacer la vue par une autre sans toucher au code.

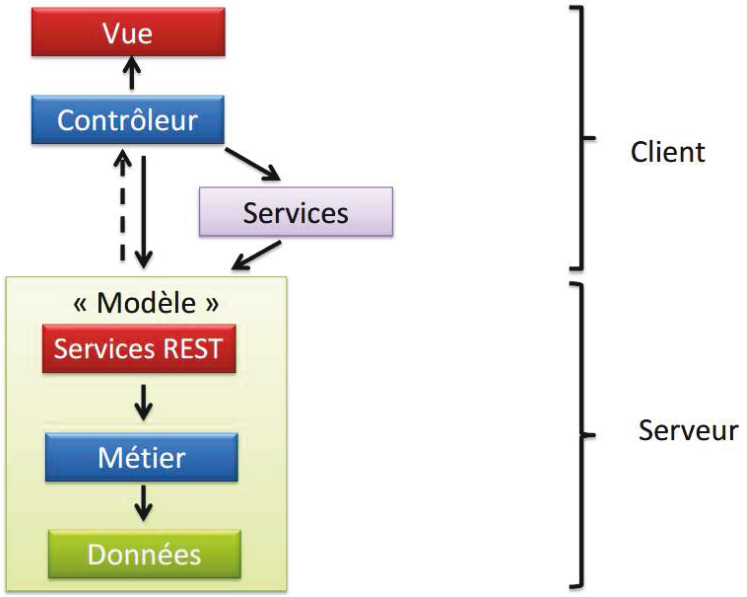


Figure 42 : FORMID est constitué de la combinaison d'un modèle MVC et d'un modèle en couches.
La partie modèle du MVC côté client peut être considérée comme étant la résultante des couches serveur permettant l'accès aux données. La vue affiche les données et renvoie les actions utilisateur, le contrôleur agit à la fois sur le modèle et la vue, en fonction des actions de l'utilisateur. Le modèle charge et stocke les données.

C'est finalement la séparation des rôles qui doit rester au cœur des préoccupations. Car dans les faits, les structures sont souvent modifiées et combinées, comme le montre la Figure 42 illustrant l'architecture de FORMID : un modèle MVC côté client lié à un modèle en couches côté serveur.

Il convient également d'évoquer l'emplacement de l'intelligence du système. La partie AngularJS côté client est une application à part entière, supportée par la possibilité de créer des services. Certains aspects fonctionnels sont donc entièrement gérés dans les contrôleurs et leurs services, côté client. La répartition dépend en fait des cas d'utilisation, dont voici les deux extrêmes :

- La gestion des dispositifs externes est entièrement pilotée par la partie *client*.
- La gestion du scénario pendant l'exécution d'un exercice est entièrement pilotée par la partie *serveur*.

L'architecture de FORMID permet l'organisation de tous les besoins techniques et fonctionnels. La partie *client web* est une application AngularJS, et la partie *serveur* une pure application Java EE, avec des services REST, la persistance des objets en base de données et le métalangage de requête.

3.1.3.2 Patrons de conception

Les patrons de conception, nommés plus fréquemment *design patterns* même en français, sont des bonnes pratiques d'architecture logicielle reconnues et éprouvées, désignant principalement le travail de quatre co-auteurs surnommés le *GoF* (« *Gang of Four* ») [**GAMMA94**]. D'autres patrons de conception se sont fait connaître depuis. Identifier les patrons de conception demande de les connaître et d'avoir une certaine habitude de leur mise en pratique. Il est toujours intéressant d'arriver à les localiser dans le code de l'application ou de ses composants, car cela aide beaucoup à comprendre la logique globale.

Voici les patrons de conception que j'ai identifiés :

- Le ***Singleton*** permet de garantir qu'une et une seule instance d'une classe existe. Il est utilisé pour la classe principale de l'application FORMID et la gestion de propriétés de l'application.
- Le patron ***Observer*** est la base de la gestion d'événements. Si l'on souhaite être informé quand des événements particuliers se produiront, alors nous devenons l'observateur et nous sommes notifiés par l'observé auquel nous sommes abonnés.
- Le patron ***Strategy*** permet de modifier le comportement d'un algorithme lors de l'exécution.
    - C'est ce qui se passe avec un scénario. Le code principal se trouve dans la classe abstraite `Scenario`, et l'implémentation est réalisée dans chaque scénario d'auteur. Le comportement dépendra donc du fichier chargé et instancié.
    - J'ai également utilisé ce mécanisme lors de l'ajout de la méthode de communication par service web comme alternative aux WebSockets. J'ai mis en place une nouvelle interface implémentée par les deux gestionnaires. Ainsi, si dans le cas des WebSockets les messages sont bien envoyés immédiatement lors de chaque appel à la méthode `sendMessage()`, dans le cas du service web ils sont stockés dans une mémoire tampon pour être tous envoyés simultanément. Je n'ai ainsi pas eu à modifier la logique de tout le code existant.
    - Je l'ai utilisé aussi dans le gestionnaire de communication avec les dispositifs externes : une même méthode a un effet différent selon le dispositif, notamment pour lire l'état du micromonde.
- Le patron ***Façade*** permet de masquer la complexité, en offrant à l'utilisateur d'un service des opérations directement utilisables dans un contexte donné. En regardant bien, chaque service REST de FORMID regroupe les méthodes utiles dans un contexte particulier (utilisateur, exercice, séance, cours…). Par exemple, la méthode de récupération des séances d'un élève va lister uniquement les séances actives des groupes dans lesquels l'élève est inscrit.
- Le patron ***Interceptor*** a pour vocation de pouvoir placer des traitements entre un émetteur et un récepteur sans avoir à changer l'un ou l'autre.
    - Il est utilisé dans FORMID côté serveur pour gérer l'aspect sécurité : ainsi avec une seule classe de gestion, l'autorisation d'accès à chaque méthode est contrôlée, et la session de l'utilisateur maintenue.
    - Ce patron est également utilisé côté client pour gérer les erreurs des requêtes au serveur. Ainsi, une réponse HTTP « *401 : Unauthorized* » va automatiquement déclencher une redirection sur la page d'identification d'un utilisateur, quelle que soit la requête à l'origine de cette erreur.

- Le patron ***Inversion of Control*** ou ***IoC*** (« inversion de contrôle » en français) est utilisé par les cadriciels fortement structurants, qui gèrent l'application et prévoient des zones pour le développeur. AngularJS effectue des traitements de manière autonome. Il a un besoin vital de reconnaître et d'interpréter les directives présentes au sein du HTML avant même que le moindre code JavaScript du développeur ne s'exécute. Ce patron nécessite l'implémentation du patron *Dependency Injection*.
- Le patron ***Dependency Injection*** ou ***DI*** ("injection de dépendances" en français), permet de déléguer la gestion des instances. Par exemple son usage permettra à la couche métier de déclarer qu'elle a besoin d'accéder à la couche d'accès aux données et d'obtenir de quoi le faire. Sans cela, les instanciations avec les paramètres qu'elles nécessitent sont diluées dans l'ensemble du programme et posent de nombreuses problématiques d'architecture. Cette notion me semble tellement importante que la partie suivante lui est dédiée.

3.1.3.3 L'injection de dépendances

Ceci est une technique avancée de développement. Elle consiste à déléguer la gestion des instances des composants logiciels. Cette notion est importante mais très pointue, nous allons donc exposer les problématiques ayant mené à l'invention de l'injection de dépendances, puis nous verrons comment elles sont utilisées dans FORMID.

Prenons l'exemple d'un programme architecturé en couches, avec une couche de présentation, une couche métier et une couche d'accès aux données comme illustré sur la Figure 43.

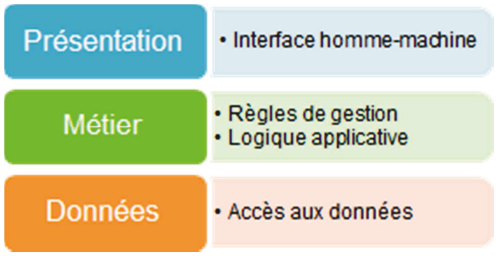


Figure 43 : architecture en couches d'un logiciel.
Source : http://www.inotekk.com/developpeur-java-j2ee-nice.php

L'objectif du développement sera d'isoler les couches pour que chacune ne connaisse que la couche directement inférieure. Les couches métier et d'accès aux données peuvent être composées de différents services. Immédiatement se pose la question de l'initialisation des couches. La couche métier doit fournir les paramètres nécessaires à la couche d'accès aux données, mais elle n'est pas censée les connaître.

L'utilisation d'un fichier de paramétrage lu par la couche d'accès aux données pourrait sembler idéale. Cependant, cette solution est très limitative, parce qu'elle n'offre pas la possibilité d'utiliser la POO (Programmation Orientée Objet) pour étendre ou modifier le comportement de la couche d'accès aux données, comme journaliser les accès ou garder une connexion ouverte.

Une solution pourrait être d'instancier une classe de la couche d'accès aux données, puis de la passer en paramètre. Cela nécessite que la couche métier fasse transiter une interface de gestion des données depuis son appelant (couche supérieure) vers la couche de données (couche inférieure). Cette méthode permet bien d'avoir le contrôle et de faire de la POO, mais produit un code lourd et qui créée des dépendances inutilement.

Enfin nous arrivons à la notion d'injection de dépendances. L'objectif est que les services nécessaires soient automatiquement transmis, avec comme prérequis une simple déclaration de leur utilisation. Cela est rendu possible grâce à un moteur d'injection de dépendances. Pour la compréhension des concepts utilisés, considérer le moteur d'injection de dépendances comme une boîte noire est suffisant.

Concrètement dans FORMID, si nous prenons comme exemple le contrôleur du module auteur « `authorController` », il est juste nécessaire d'indiquer que pour fonctionner il a besoin des services de multilinguisme « `localize` » et d'accès aux services REST « `Restangular` ». Les services seront alors automatiquement passés en paramètre lors du démarrage du contrôleur.

D'une manière générale, nous n'avons plus à nous soucier des instances des services.

L'injection de dépendances est donc très utile pour obtenir un code modulaire, évolutif, architecturé, tout en restant clair et concis.

3.2 ENVIRONNEMENT TECHNIQUE

3.2.1 SERVEUR

Le serveur est un Linux CentOS 6.6 (x86_64) distribution LTS (Long Time Support), stable jusqu'en 2021.

- Il fait tourner le serveur Wildfly qui gère les applications Java EE et écoute sur le port 8080.
- Il intègre le serveur Apache 2.2.15 qui écoute sur le port 80. Les WebSockets ont nécessité l'ajout d'un module complémentaire, car elles ne sont gérées nativement qu'à partir de la version 2.4, mais ce n'est plus un problème car les WebSockets ont été mises en sommeil.
- Il héberge la base de données PostgreSQL version 8.4.20 incluse dans le LTS, ainsi que la version 9.4.1 utilisée pour FORMID. Cette dernière était déjà installée et paramétrée lorsque j'ai appris que le système Linux était stable et prévu pour ne pas évoluer jusqu'en 2021. J'ai choisi une version récente parce que les documentations trouvées sur internet ne correspondaient pas à la version installée.

Machine virtuelle

Le serveur Linux est une machine virtuelle (VM) gérée par Jean-Claude Mas.

Je lui ai demandé pour l'inauguration Amiqual4home, du 05 juin 2015, s'il était envisageable d'embarquer la machine virtuelle sur un PC. Il avait testé et confirmé cette possibilité. Cela pourrait servir pour créer une machine de démonstration qui n'ait pas besoin du réseau.

J'estime qu'il serait intéressant de prévoir une duplication de la machine virtuelle, pour créer un environnement de validation et un de production. En effet, avec la mise à disposition du logiciel dans des écoles, il ne sera plus possible de mettre à jour le serveur de manière insouciante. Or à l'heure actuelle la seule manière qu'ont Anne Lejeune et Viviane Guéraud de tester l'application, donc de valider mes développements, est de se connecter à l'unique serveur Linux.

Configuration réseau

Le serveur est accessible à l'adresse « `formid.imag.fr` **»**. Cette machine virtuelle est administrée par des scripts automatiques Puppet, qui écrasent toutes les demi-heures certaines configurations que nous pourrions faire manuellement, notamment en ce qui concerne le réseau : « `iptables` » et « `ip6tables` ». Les routeurs de l'imag n'autorisent que l'entrée des ports : 80 (HTTP, pour le web), 443 (HTTPS, pour le web Sécurisé), et 22 (SSH pour l'administration à distance), ce sont donc les seuls ports accessibles depuis l'extérieur du réseau virtuel (VLAN). De plus, une application qui voudrait écouter sur un numéro de port inférieur à 1024 devrait être `root` (avoir tous les droits), ce qui est déconseillé.

Ces raisons justifient le fait que le serveur Apache écoute sur le port 80, et qu'il relaye, grâce à une configuration adéquate, les requêtes vers le serveur Wildfly sur le port 8080. Cette architecture déjà en place m'a été conseillée par M. Mas, ce qui assoit sa pertinence.

3.2.2 CLIENT

Les évolutions de FORMID m'ont amené à ajouter de nouveaux composants, par rapport à l'état initial décrit Tableau 2 (page 14). Le Tableau 5 liste ces composants et explique la raison de leur utilisation.

Tableau 5 : composants logiciels ajoutés côté client

Nom du composant	Usage
CodeMirror [**CODEMIRROR15**]	Est utilisé pour la coloration syntaxique du code. En effet, les conditions d'évaluation d'un scénario élaboré par l'auteur sont écrites en langage Java, et ce module aide beaucoup à la lisibilité.
Moment.js [**MOMENT.JS15**]	Cette librairie utilitaire permet de gérer très facilement les dates et les délais en JavaScript. Elle est par exemple utilisée pour calculer quel est l'âge de la dernière notification d'un dispositif externe.
ResponsiveVoice [**RESPONSIVEVOICE.JS15**]	Cette librairie JavaScript permet une synthèse vocale. Elle est utilisée dans le module élève pour lire les consignes et les différents messages de l'exercice.

Ces composants fonctionnent de manière isolée, c'est-à-dire que le fait de les intégrer à FORMID n'a pas le moindre impact tant qu'ils ne sont pas invoqués.

3.2.3 OUTILS UTILISÉS

Certains outils importants ont été nécessaires pour mener à bien les tâches qui m'ont été confiées.

Tout d'abord, il m'a fallu un comparateur de fichiers texte, pour résoudre des problèmes de mise en route détaillés dans le chapitre 1.

J'ai eu besoin d'un décompilateur Java pour pouvoir analyser le code des scénarios. Ce code est généré par la transformation du scénario XML en Java grâce à un fichier XSL, puis compilé en fichier ".class".

J'ai également utilisé un décompilateur Flash, pour analyser le code source de *TPElec*, et ainsi comprendre quelles étaient ses fonctionnalités, et ses limites. J'ai eu besoin suite à cela d'un testeur de « Sockets », qui consiste à mettre un port TCP/IP sur écoute. Je l'ai utilisé pour vérifier si *TPElec* envoyait bien des traces sur le port 1024 comme le laissait entrevoir son code.

J'utilise souvent les outils de SysInternals Suite [**SYSINTERNALS15**]. Dans le cas de FORMID j'ai eu besoin du logiciel permettant de trouver le processus propriétaire d'un port ouvert, pour arrêter le serveur en question. Mais ce besoin ne se fait plus sentir depuis que Wildfly est intégré à NetBeans et que H2 a disparu.

J'ai dû obligatoirement utiliser un client SSH pour me connecter au serveur Linux, qui sur Windows est PuTTY.

Un client FTP a également été requis pour transférer les fichiers à déployer depuis mon poste de développement vers le serveur Linux : le fichier WAR de l'application, les fichiers de traduction, mais aussi pour d'autres usages.

Pour suivre l'exécution de FORMID, il est très utile d'avoir un outil de visualisation de fichiers journaux. Ces outils ont la particularité d'afficher immédiatement toutes les nouvelles lignes. Sur Linux c'est la commande `tail`, et sur Windows j'ai opté pour le logiciel BareTail.

Et au final, l'outil le plus utile au quotidien, est le navigateur internet lui-même. Les navigateurs Google Chrome et Mozilla FireFox contiennent des outils de développement absolument fabuleux, avec points d'arrêts, pile des appels, valeurs des variables, traces réseau.

3.3 ETAT DES LIEUX DE FORMID

Le développement d'une application s'opère au niveau technique dans le but d'obtenir des résultats fonctionnels. Ce sont ces derniers qui sont jugés par le client. Les aspects techniques, invisibles par nature, peuvent néanmoins représenter de grands enjeux ou des défis complexes. Il apparaît donc opportun d'en rappeler les améliorations les plus importantes.

L'environnement de développement a été simplifié

Tout d'abord, l'environnement de travail quotidien ne comporte plus de fichiers de commande à exécuter, qui permettaient le lancement et l'arrêt du serveur applicatif Wildfly et de la base de données H2. L'astuce consistait à démarrer le serveur Wildfly manuellement, puis à demander le déploiement depuis NetBeans. Le serveur JBoss ne pouvait pas démarrer puisque le port venait d'être réservé, mais le processus de déploiement se poursuivait, et déployait en réalité sans le savoir dans le serveur Wildfly. Le fonctionnement de l'ensemble manquait cependant de stabilité et posait des problèmes de blocage de ports. Maintenant, il suffit d'ouvrir l'environnement de développement et de lancer l'application. D'une part Wildfly fait partie de NetBeans grâce à un module d'extension, et d'autre part la base de données est dorénavant hébergée par un service PostgreSQL. Ce dernier démarre automatiquement, aussi bien sur Windows que sur Linux.

La base de données a été migrée sur PostgreSQL

Le succès de la suppression de la base de données H2 a été un réel soulagement. Comme H2 est un programme, la gestion de ses instances était complexe et la localisation de ses fichiers également. La moindre modification de base de données demandait des manipulations très longues pour rapatrier la base en local, pour la modifier et la renvoyer. L'administration des bases de données locale et distante s'effectue maintenant très confortablement sur l'interface web phpPgAdmin (Cf. Annexe 9).

Le mode élève n'utilise plus les WebSockets

J'étais réticent à l'idée de supprimer les WebSockets à cause de la perte de capacités de communication engendrée. Nous avions d'un autre côté en permanence des problèmes bloquants pour le module élève. Avec l'approche des installations en classe, il fallait donc trouver une solution. Leur remplacement par des services web a insufflé un sentiment de fiabilité.

FORMID communique avec les dispositifs externes de manière asynchrone

Il était nécessaire pour le dispositif tangible, et pour les autres dispositifs externes qui ont suivi, de faire évoluer le système de communication créé pour *TPElec*. Il paraissait incontournable de mettre en place des échanges asynchrones avec le dispositif tangible afin de ne jamais bloquer l'interface utilisateur.

L'auteur peut tester le scénario dans l'onglet d'exécution

Ceci est une amélioration fonctionnelle, certes, mais l'aspect technique représentait un véritable défi : comme une application AngularJS ne contient qu'une seule page web, afficher le dispositif externe à la fois dans l'onglet conception et dans l'onglet exécution revenait à dupliquer le code source du dispositif externe dans la même page. J'ai donc été confronté à bon nombre de problèmes de gestion de ces composants.

L'adaptateur de communication de chaque dispositif externe est un contrôleur AngularJS

Chaque adaptateur de communication, qui sert d'interface entre FORMID et le dispositif externe, était originellement constitué de méthodes JavaScript globales à l'application (par exemple avec "`window._getVariables = function() {}`") et invoquées uniquement de manière synchrone. Sous la pression de multiples contraintes, ce système a subi plusieurs évolutions : tout d'abord il a été placé dans une classe JavaScript, qui a ensuite beaucoup évolué, avant que ne soit osée la tentative de le transformer en AngularJS. La réussite n'était pas certaine car le code JavaScript du contrôleur est intégré à la page web de l'adaptateur de communication, qui elle-même est chargée dynamiquement. Grâce à la mise en place des contrôleurs AngularJS, j'ai pu créer un service de gestion des dispositifs externes, « `SimulationProvider` », qui dialogue avec eux au moyen d'événements. Ce système est extrêmement pratique et adaptatif. D'une part, il permet de gérer les promesses en cascade, et d'autre part, si un adaptateur de communication ne gère pas une méthode, le service de gestion s'en accommode sans problème. De manière générale, l'interface de communication entre FORMID et les dispositifs externes présents et futurs est dorénavant architecturée, générique et évolutive.

Il est possible de modifier l'état du dispositif externe à tout moment

D'une part les contraintes techniques introduites dans FORMID par la spécificité de fonctionnement de *TPElec* ont été levées (partie 2.2.2). Et d'autre part, la création du gestionnaire de dispositifs externes, et le fait que chaque adaptateur de communication soit dans un contrôleur AngularJS, ont permis une grande évolutivité tout en conservant l'aspect générique. Les dispositifs externes fonctionnent maintenant de manière unifiée.

3.4 MODÈLE DE DONNÉES

Le modèle de données va être présenté par parties pour plus de lisibilité. Nous allons commencer par les cours, puis nous verrons les traces tuteur, pour terminer avec les traces élève, auteur, les rôles et les praxéologies.

3.4.1 DIAGRAMME PRINCIPAL

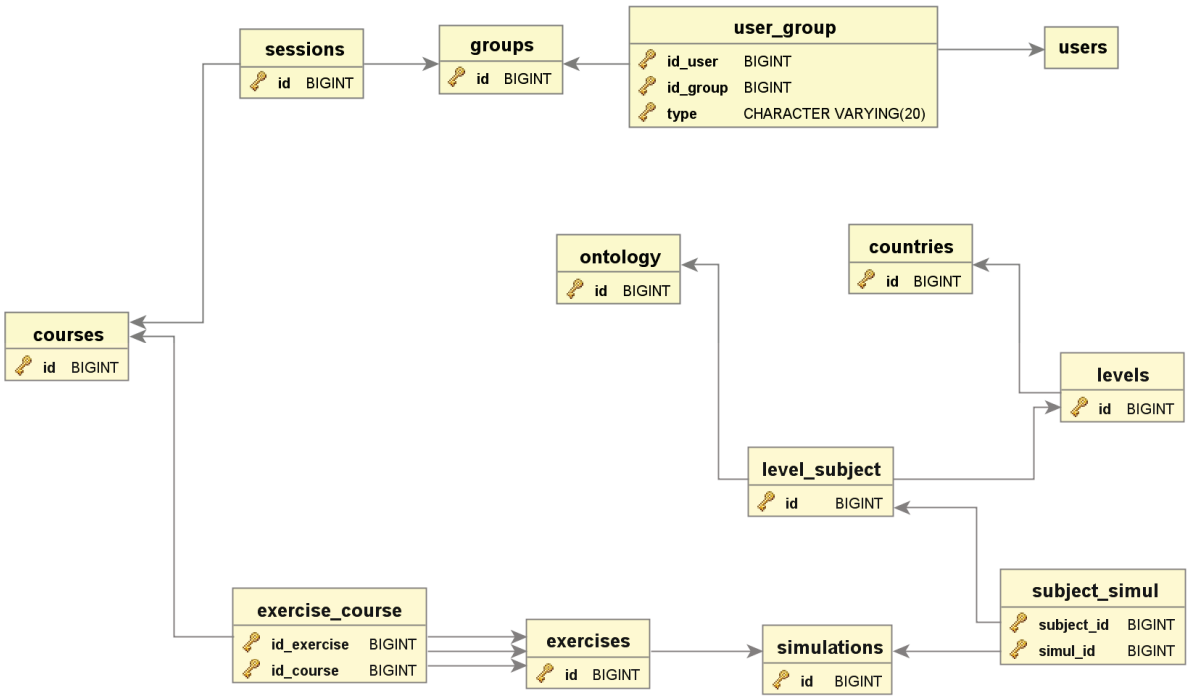


Figure 44 : diagramme de la base de données, vue principale.

Le cœur du logiciel est représenté par le diagramme Figure 44.

- Une séance de travail `("sessions")` est donnée pour une date et concerne un cours `("courses")` et un groupe d'élèves `("groups", "user_group", "users")`.
- Un cours `("courses")` contient des exercices `("exercise_course", "exercises")` qui peuvent porter sur différents dispositifs externes `("simulations")`.
- Un dispositif externe `("simulations")` ne peut être utilisé dans un scénario que s'il est adapté à la discipline et au niveau scolaire du scénario `("level_subject", "levels", "countries")`.
- À une discipline et un niveau scolaire donnés correspondent un serveur de praxéologies `("ontology")`.

Pour compléter ces principes, voici des informations complémentaires :

- Les 3 liens entre "`exercise_course`" et "`exercises`" sont justifiés par le fait qu'au sein de chaque cours, chaque exercice est chaîné au suivant, et qu'il est possible de bifurquer vers un exercice différent si l'élève échoue à l'exercice courant.
- Le serveur de praxéologies (`"ontology"`) peut être le même pour différentes disciplines et différents niveaux, parce que lors de la requête à ce serveur, nous fournissons en paramètre le niveau et la discipline.

3.4.2 DIAGRAMME DES TRACES TUTEUR

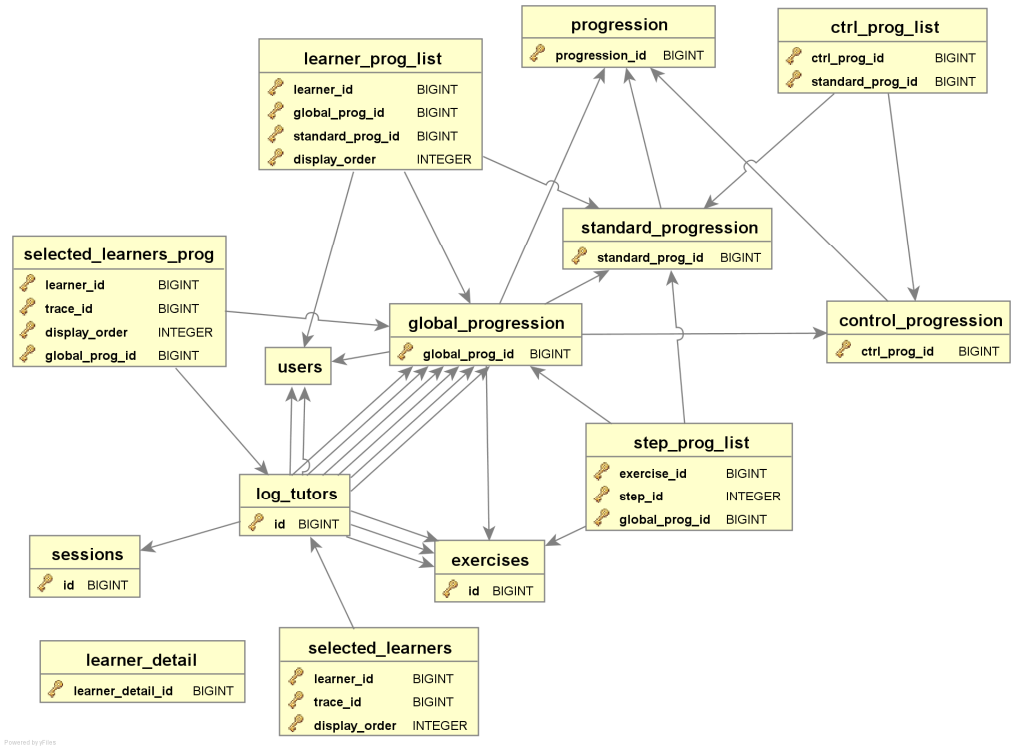


Figure 45 : diagramme de la base de données, traces tuteur

Le diagramme Figure 45 concerne uniquement les traces du tuteur. Pour tenter d'expliquer, j'ai déduit une liste de faits par l'analyse du code source et des données de la base. Cette liste m'a permis de créer la Figure 46.

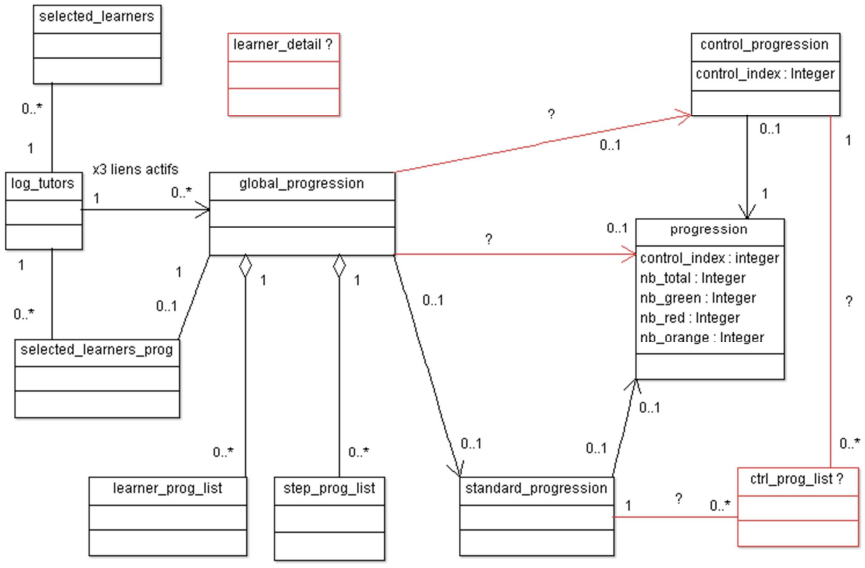


Figure 46 : diagramme UML des traces tuteur établi par rétro-conception.
Les cardinalités à zéro avec un lien rouge indiquent en fait une cardinalité « 0..1 » qui n'est jamais renseignée en base de production.
Les entités en rouge et avec un point d'interrogation ne contiennent aucune donnée.

Ce diagramme clarifie un peu la compréhension de l'ensemble, mais il faudrait cependant continuer la rétro-conception. Cette analyse montre que le modèle de données des traces tuteur est à l'heure actuelle encore sous-exploité. Il a certainement été conçu pour stocker plus d'informations qu'il n'en gère actuellement.

3.4.3 DIAGRAMME DES ÉLÉMENTS SECONDAIRES

Outre le diagramme des éléments principaux, et celui très riche des traces tuteur, les autres notions peuvent être analysées rapidement.

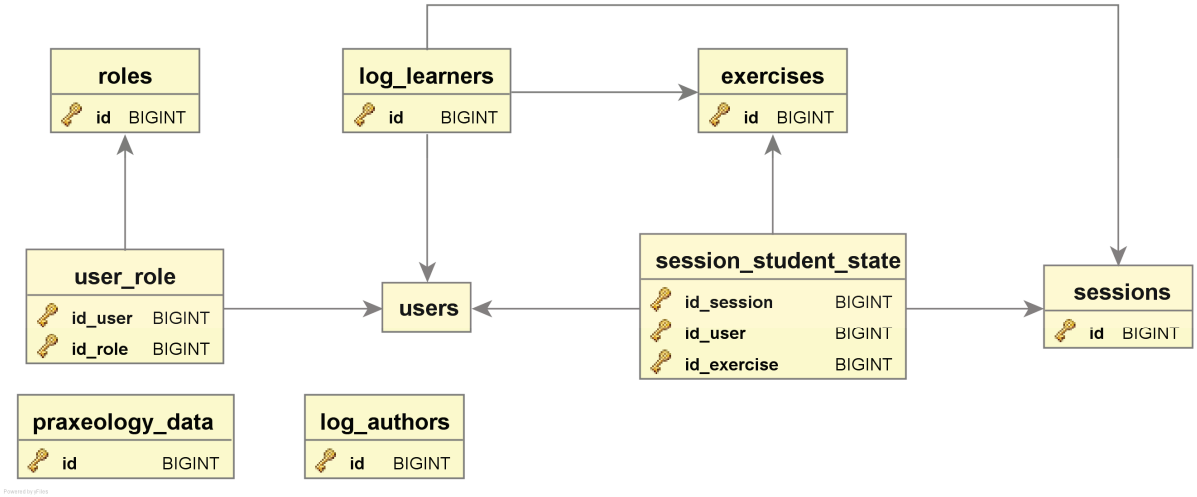


Figure 47 : diagramme de la base de données : traces auteur et élève, rôles, praxéologies

Les éléments représentés Figure 47 peuvent se comprendre ainsi:

- Un utilisateur ("`user`") peut avoir plusieurs rôles ("`user_role`"). Les rôles principaux ("`role`") sont élève, tuteur, auteur, administrateur.
- L'avancement d'un élève ("`session_student_state`") sur un exercice ("`exercises`") pendant une séance de travail donnée ("`sessions`") se matérialise par les états suivants : démarré, réussi, échoué. La couleur des exercices que l'élève voit dépend de cette information.
- Les traces auteur ("`log_authors`") enregistrent les actions de l'auteur afin d'analyser a posteriori sa méthode de création de scénario. Cet aspect devrait être enrichi.
- Les traces de l'élève ("`log_learners`") décrivent l'activité de l'élève, en termes de contrôles de situations observables déclenchés et de résultats des demandes de validation d'étape.
- Le fac-similé de praxéologies consistait en une liste de données JSON préparées ("`praxeology_data`") qui étaient lues et renvoyées depuis un serveur local pour mimer le fonctionnement du serveur réel. Cette table est devenue obsolète fin novembre 2015, avec le développement de notre propre serveur de praxéologies qui a sa base de données dédiée, décrite ci-après.

3.4.4 DIAGRAMME DE LA BASE DES PRAXÉOLOGIES

Le serveur de praxéologies développé par le groupe de travail FORMID possède sa base de données propre représentée Figure 48. L'ensemble des tables servent donc le but de fournir les données praxéologiques via service invoqué par FORMID.

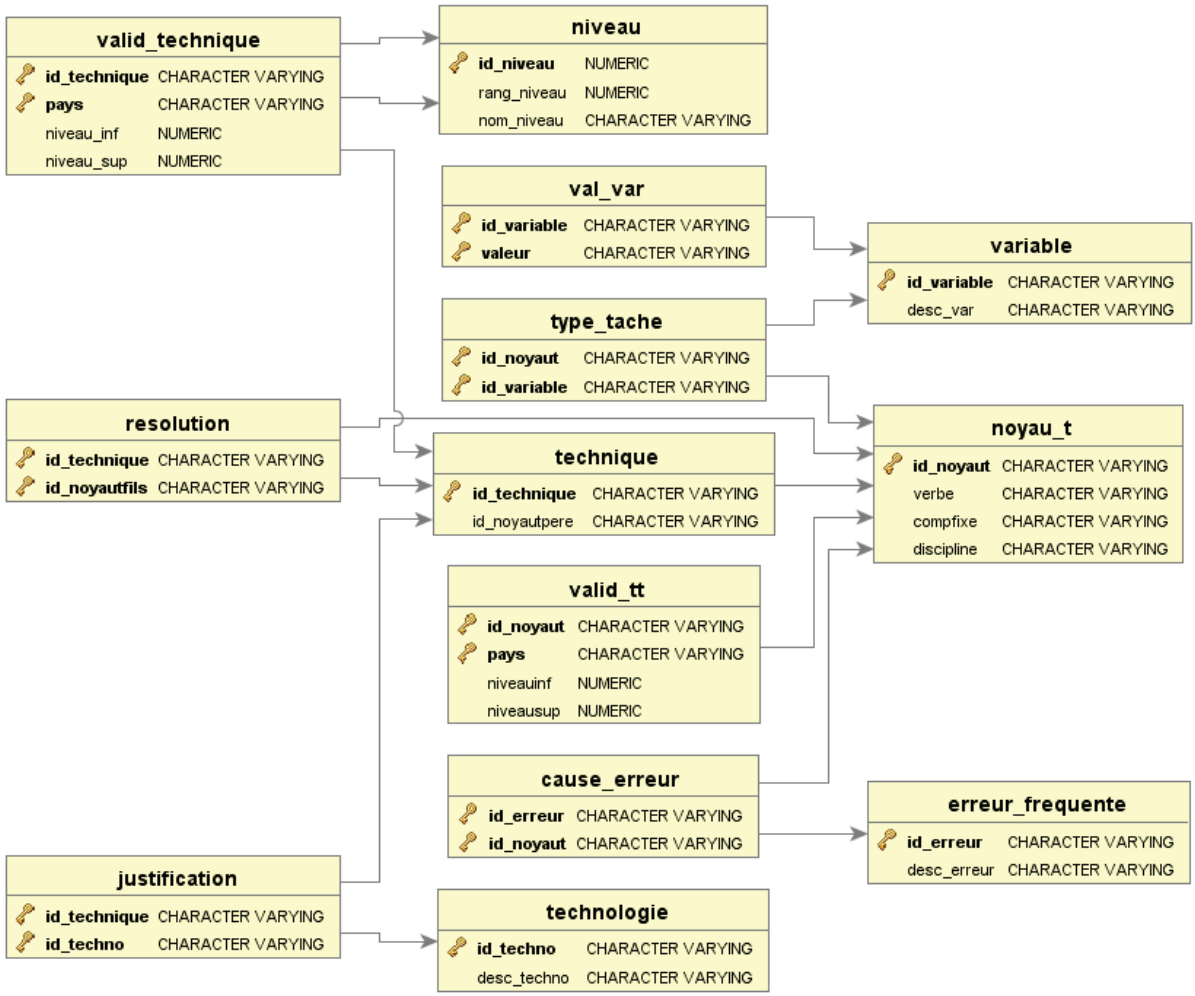


Figure 48 : diagramme de la base de données des praxéologies

- Un type de tâche ("`type_tache`" et "`noyau_t`") est résolu par une ou des techniques ("`technique`", "`resolution`") qui se décomposent potentiellement elles-mêmes en types de tâches.
- Une technique de résolution est valable pour un niveau scolaire et un pays donnés ("`valid_technique`"). Un type de tâche est valable selon des critères équivalents ("`valid_tt`").
- Un type de tâche peut avoir des facteurs d'influence ("`variable`") qui peuvent prendre plusieurs valeurs ("`val_var`").
- Une erreur fréquente ("`erreur_frequente`") peut être rencontrée dans plusieurs types de tâches ("`cause_erreur` ").
- Une technique ("`technique`") est justifiée ("`justification`") par une ou plusieurs technologies ("`technologie`").

3.5 GESTION DE L'INFORMATION

L'information est importante, mais trop souvent négligée. En tant qu'intervenant, les traces que nous laissons se font uniquement par écrit, à travers les documentations, les commentaires dans le code, et le mémoire.

3.5.1 DOCUMENTATIONS PRODUITES PAR MON TRAVAIL

De la même manière que j'ai l'habitude de prendre des notes pour garder une trace des explications que l'on me donne, je consigne tout ce que je découvre. Cela me paraît indispensable. D'une part parce que je peux retrouver sans peine toutes les informations précises sans avoir à les chercher à nouveau, et d'autre part parce que si j'avais eu ces informations, j'aurais gagné du temps de recherche. Je laisse donc des traces pour que les suivants et suivantes ne buttent pas sur les mêmes écueils. Au fil des 9 mois de stage, j'ai donc beaucoup rédigé. Il n'y a jamais eu de phase de rédaction, mais des documents créés au fur et à mesure. Afin d'être plus concret, le tableau en Annexe 10 contient la liste exhaustive des documents que j'ai rédigés. Le volume total des 50 documents représente 300 pages, ce que je considère comme conséquent.

Les documents peuvent être classés en catégories :

- Les entrées de la base de connaissance.
- Les aide-mémoire sur un sujet spécifique.
- Les études, propositions et spécifications avant réalisation.
- Les documentations sur ce qui a été produit.
- Les documents de travail temporaires.

Les documents de type aide-mémoire servent à regrouper les informations liées. Par exemple celui concernant le déploiement indique quels sont les éléments à copier sur le serveur Linux et dans quels répertoires. Une entrée de la base de connaissance contient trois rubriques que j'ai empruntées à Microsoft : le symptôme, la cause, la résolution. Enfin le document principal est celui destiné au nouvel arrivant, qui est un guide et qui référence la plupart des autres documents.

3.5.2 COMMENTAIRES DANS LE CODE

3.5.2.1 Introduction

Les avis rencontrés divergent fortement sur le sujet. Deux courants de pensée qui peuvent être souvent extrêmes. D'un côté les gens qui sont pour le commentaire, et de l'autre ceux qui militent pour un code sans commentaire ([**FRAMABLOG15**]). Voici des avis qui vont à l'encontre de ce que je pense :

"Approprie-toi ce code sans commentaires pour savoir si tu es un bon développeur."

"Alan Cox (bras droit de Linus Torvald, créateur de Linux), affirme qu'un bon programme n'a pas besoin des commentaires, puisque sa lecture doit être fluide et propre pour qu'il se comprenne sans (commentaire)."

"N'insultez pas l'intelligence de vos successeurs."

Dans la même lignée, [**CODING HORROR15**] (en anglais) explique que si l'on éprouve le besoin de mettre un commentaire, c'est certainement parce que le code est mauvais. Certaines analyses comme [**MIXIMUM15**] sont plus modérées et tentent de comparer les arguments des uns et des autres.

Je voulais montrer qu'il n'y a aucun consensus sur le sujet, avant d'exposer ma manière de penser et de faire.

3.5.2.2 Pourquoi je commente

Je pense être quelqu'un de très efficient dans mon métier, mais je constate souvent que je pourrais économiser un temps précieux s'il y avait une simple ligne de commentaire. Dans un contexte professionnel, je suis en grande majorité confronté à des logiciels de gestion à longue durée de vie. Je ne peux pas exclure la possibilité qu'une conception parfaite éviterait les situations d'intenses interrogations. Mais il n'est pas concevable d'exiger que chacun produise un code exemplaire pour des raisons évidentes de compétences, de coût et de contraintes de l'existant. Commenter permet de faciliter la maintenabilité du code à moindre frais.

Le commentaire doit expliquer en premier lieu pourquoi le code est là

Certes, un morceau de code bien écrit va montrer qu'il enregistre les données reçues par le serveur dans un fichier. Mais si un commentaire explique que ce fichier provient d'un dispositif externe et qu'il sera en attente jusqu'à ce "FORMID client" demande à obtenir son état, cela devient limpide. Le « pourquoi » de la raison d'être d'un morceau de code permet à l'intervenant de l'inscrire mentalement dans le processus fonctionnel.

Le commentaire doit expliquer ce que le code est censé faire

Le commentaire sert de mini-spécification locale. En effet, comment savoir si une valeur renvoyée dans un cas limite est normale ou indésirable ? Il m'est fréquemment arrivé dans ma carrière que je me questionne sur ce qu'avait voulu faire le créateur, sur ce que le code devrait idéalement renvoyer. Un petit morceau de code peut faire perdre des journées de recherche.

De plus, il peut être ardu de se rendre compte que l'action du code n'est qu'un moyen, et pas un besoin en soi. Pour reprendre l'exemple du fichier en attente, un autre intervenant pourra décider de modifier la méthode, et conserver les données dans la mémoire. Il n'a pas à rechercher où et dans quels cas d'utilisation ce fichier est lu pour comprendre que le stockage dans un fichier n'est que la manière que j'ai choisie pour résoudre mon problème.

Comment j'ai procédé

À chaque fois que j'ai à intervenir sur une ligne ou un bloc de lignes, je note mes initiales et la date entre parenthèses, précédés par une explication. Par exemple :

```
found = nodeVar; // - changement pour renvoyer le noeud de la variable et pas "true" (ELR 20/07/2015)
```

Mon trigramme est donc naturellement présent un peu partout, le programme en comporte 2071 occurrences.

3.5.3 PERSPECTIVES

La pertinence, ou non, de la prolifération de mes documentations, pourra être appréciée par les intervenants qui me succéderont sur le projet. J'ai bon espoir que cela les aidera à être plus vite opérationnels et efficients. Il m'intéresserait d'en être informé pour faire évoluer mes pratiques.

3.6 CONCLUSION DU CHAPITRE

Le chapitres précédents ont traité de l'immersion dans le sujet, puis des réalisations effectuées, pour enfin passer en revue dans le présent état des lieux toutes les informations qui pourraient être utiles pour comprendre où en est FORMID. Le déroulement chronologique est maintenant terminé, il est temps de prendre du recul pour passer en revue l'analyse de l'ensemble de l'activité.

4 MAÎTRISE

Maîtriser un projet demande de savoir prendre du recul pour observer et gérer, puis d'être capable d'en tirer des enseignements. Je vais donc exposer ma progression depuis ma volonté initiale de gestion de projet, jusqu'à la méthode finalement employée. Puis je ferai état de différents aspects de la gestion de projet appliqués à FORMID. En seconde partie, je ferai le bilan des données en ma possession.

4.1 GESTION DE PROJET

4.1.1 INTRODUCTION

Dans mon cursus de développeur, je n'ai jamais eu l'opportunité de gérer un projet. Mon objectif, outre la mission qui m'a été confiée, était de profiter de mon travail au sein du LIG pour mettre en place une gestion de projet. Cette partie présente ma démarche et décrit donc quels sont les aspects qui ont pu être maîtrisés.

En premier lieu, la raison d'être de la gestion de projet consiste à dominer son travail, par une vue d'ensemble et une connaissance du détail à tout instant. Cette gestion sert aussi à suivre l'évolution, et à s'assurer de l'atteinte des objectifs fixés dans les délais impartis. Il s'agit de mettre en corrélation les exigences de la maîtrise d'ouvrage (MOA) avec les moyens mis à la disposition de la maîtrise d'œuvre (MOE).

4.1.2 CHOIX DE L'OUTIL

Pour mener à bien mon désir de gestion de projet, je me suis grandement basé sur l'ouvrage « *Management d'un Projet Système d'Information* » [**MORLEY08**] qui m'a été utile sous bien des aspects, et que je nommerai par la suite "ouvrage de référence". J'ai également beaucoup utilisé internet comme source d'informations.

Dans l'optique de gérer tous les aspects imaginés, j'ai commencé par vouloir consigner toutes les informations sous forme exploitable. C'est donc sur cette base que j'ai cherché un logiciel qui me permettrait de les saisir et de les traiter. Afin de ne pas pénaliser le travail sur FORMID par ma volonté masquée de gérer mon projet, le temps qui y a été consacré a été réparti sur les premières semaines. Il n'y a pas eu de phase rigoureuse de comparaison des outils, mais des essais, dans l'espoir de trouver rapidement un produit adapté.

4.1.2.1 Première tentative

Parmi les outils de planification et de suivi *open source* que j'ai identifiés, j'ai commencé par Gantt Project parce qu'il avait été utilisé par mon prédécesseur et qu'il est mentionné dans mon ouvrage de référence. La Figure 49 montre la traduction de ce premier travail.

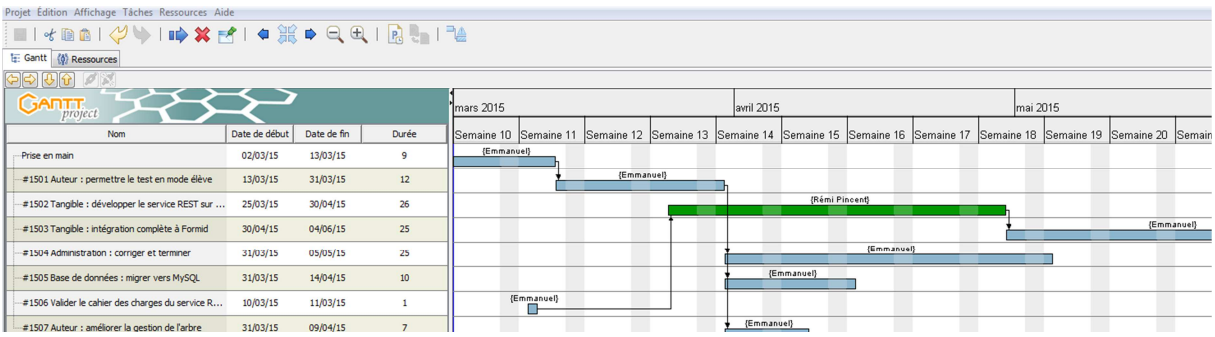


Figure 49 : début de gestion du projet avec GanttProject

Malgré l'apparence visuelle très agréable, les résultats obtenus ne me convenaient pas à plusieurs égards. Tout d'abord, j'aurais aimé pouvoir intégrer des dates « au plus tard » afin de visualiser un rétro-planning qui se termine à la date prévue de mise en production. Ensuite, malgré l'identification de la même ressource « Emmanuel », les différentes tâches sans relation d'antériorité sont affichées de manière simultanée. J'aurais aimé voir une proposition d'agencement des tâches secondaires qui ne remettrait pas en cause le chemin critique. Je n'ai pas été non plus convaincu par les possibilités de saisie des plus petites tâches, puisque la durée est en jours et que la création de nombreuses tâches nuirait à la lisibilité de l'ensemble.

J'ai donc constaté que je n'avais pratiquement aucun besoin de la gestion d'antériorité, donc de l'analyse du chemin critique, et que je ne désirais pas seulement planifier, mais également suivre toutes les tâches réalisées. En conclusion, j'ai considéré que ce logiciel n'était pas du tout adapté pour moi.

4.1.2.2 Recherche éclairée

J'ai donc recherché un logiciel qui me permettrait d'obtenir un tableau de bord de gestion de projet, et dont l'unité est le jour-homme pour avoir une vue d'ensemble de l'avancement. Mais un logiciel qui serait aussi basé sur une saisie fine des tâches et des temps consommés en heures. J'ai également besoin dans mon rôle de développeur de conserver une vision « journal » de tous les temps consommés, avec la possibilité de contrôler le total par jour et par semaine. Cela permet de localiser une erreur ou un oubli de saisie. J'aimerais aussi ne pas être restreint à des tâches, pour pouvoir saisir des réunions, des installations de logiciels ou toute autre activité réelle.

J'ai donc testé plusieurs outils de gestion de projet gratuits, dont les captures d'écran sont en Annexe 11.

Open Workbench (Annexe 11.1) est libre et gratuit. Il doit être installé sur le poste de travail. Il existe en deux versions, communauté ou entreprise. Cependant, la page Wikipédia indique que le projet n'est plus maintenu par la communauté depuis 2008 et que le site est indisponible depuis "des mois". Il permet de gérer à la fois les projets, les sous-projets, les phases, les tâches et les activités dans un tableau de Gantt. Il s'apparente beaucoup à Gantt Project, qui ne me convenait pas en termes de fonctionnalités.

ProjeQtOr (Annexe 11.2) est un logiciel gratuit et *open source* qui a de plus le mérite d'être en français. Il semble très riche fonctionnellement, puisqu'il permet de gérer un portefeuille de projets. C'est un logiciel qui requiert visiblement une formation conséquente. La démonstration en ligne permet de se connecter comme superviseur, gestionnaire (chef de projet) et membre. Le logiciel fonctionne avec une notion d'activité qui peut être par exemple une tâche ou du management. Les imputations de temps consommés sont saisies directement par les membres, en jours. D'après les concepteurs, ce logiciel est conçu pour gérer uniquement des tâches d'une granularité de 1 à 10 jours. Donc inadapté aux petites tâches qui jalonnent le quotidien.

Project Open (Annexe 11.3) gère 8 profils différents, dont notamment le directeur, le chef de projet et l'employé. Il permet de gérer de multiples projets simultanément. Il est possible de saisir ses heures passées pour des tâches. L'intervenant a une vue sur sa semaine, et peut valider le total de ses heures travaillées (Cf. Figure 50). Cette validation entraîne la création d'un processus qui vise à faire approuver le temps consommé par le responsable. Ce dernier peut consulter le détail des heures effectuées avant de les valider.

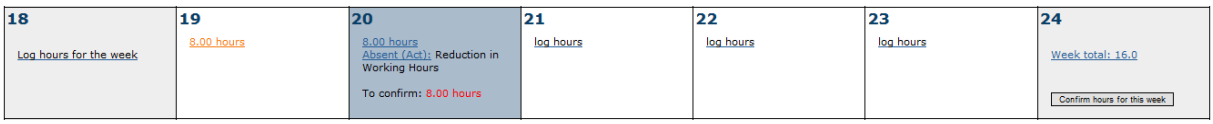


Figure 50 : vue d'ensemble des heures d'une semaine de travail sur Project Open

Ce logiciel répond donc à une partie des besoins. Cependant, tout temps saisi doit être associé à une tâche, et par conséquent, par exemple pour des réunions, soit je ne les trace pas, soit elles deviennent des tâches.

L'absence d'une demi-journée que l'on voit au 20 octobre n'entre pas dans le compte des heures, ce qui rend à mon sens plus difficile la vérification de la saisie. De plus, la navigation pourrait être grandement facilitée par des liens entre les écrans. Enfin, bien que chaque temps soit affecté à une tâche, il n'est pas possible de saisir le type d'activité (spécification, développement, formation…). Je trouve dommage de perdre cette information.

Project.Net (Annexe 11.4) est un outil web *open source*. Il ne dispose pas d'une version d'évaluation en ligne, mais seulement d'une vidéo. Après le téléchargement, l'installation est manuelle. Les prérequis sont un serveur Apache Tomcat, mais surtout une base de données Oracle. Dans ces conditions, un test rapide n'est pas possible, il est nécessaire de lui consacrer bien plus de temps.

J'ai dans les faits à peine effleuré chaque logiciel, puisque je voulais trouver très rapidement une solution intuitive et adaptée à mes besoins.

4.1.2.3 Solution du tableur

Je n'ai pas trouvé de logiciel simple qui me permette de saisir mes temps d'un point de vue de développeur, et qui les ventile automatiquement dans les tableaux de bord. Il aurait peut-être fallu y consacrer plus de temps. Cependant, je commençais à consommer beaucoup de temps dans les semaines 4 et 5 (Cf. Figure 51), et surtout, plus le temps passait et plus j'avais de tâches à reporter dans le gestionnaire de projet. Je devais surtout, indépendamment de cela, monter en compétences sur FORMID pour devenir opérationnel.

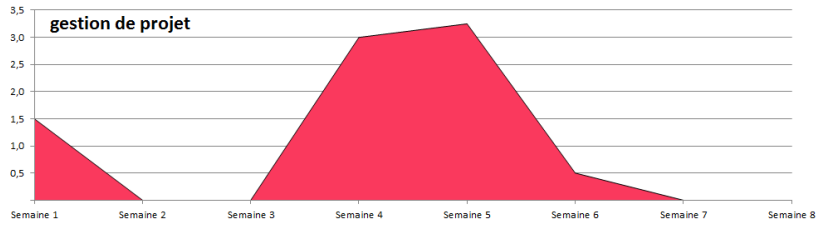


Figure 51 : temps consacré à la gestion de projet les premières semaines.

Afin de ne pas perdre d'informations pendant la phase de choix d'un outil de suivi, je consignais toutes les données dans un tableur. Puis j'y ai ajouté le tableau de bord représenté Figure 52. Il permet de saisir le temps consommé (noté C), le temps restant estimé (noté R), et le calcul de l'avancement (noté A).

			16/03/2015			23/03/2015			30/03/2015			06/04/2015		
			Semaine 3			Semaine 4			Semaine 5			Semaine 6		
Lot	Tâche	Estimation (j)	C	R	A	C	R	A	C	R	A	C	R	A
	#1500 Corrections d'anomalies et améliorations mineures		2,5			2,1375			1,34					
	#1501 Auteur : permettre le test en mode élève	12		12		2,1375	7	5						
	#1502 Auteur : restaurer le modèle vide et le préserver de l'écrasement	2												
	#1503 Auteur : rendre utilisable le module (gestion de session, publication de scénario, compilation, affichage)	2										1,95		
	#1504 Migrer de H2 vers MySQL ou postgreSQL	10		10	0		10	0	2,25	4	6	0,3125	4	0
	#1505 Corriger la gestion des sessions : connectedUsers est "static" et ne doit pas gérer les fermetures sans déconnexions	3												
4.4 - MODE TUTEUR														
4.5 - MODE ELEVE														
	#1506 Correction d'anomalie de communication WebSockets								0,5	1		0,5		
	#1507 Corrections d'anomalies diverses											0,125		
4.9 - DOCUMENTATION														
Absence														
	Congé											1		
	Maladie													
	Non travaillé											1		
	Somme de contrôle 5 jours par semaine		5,0			5,0			5,0			5,0		

Figure 52 : tableau de bord en jours-homme créé pour gérer le projet FORMID.
Il contient les jours consommés (C), les jours restants (R) et l'avancement (A) par tâche et par semaine. Modèle suivi : [**MORLEY08**].

J'ai maintenu ce tableau de bord à jour pendant quelques semaines. Puis je suis finalement arrivé au constat qu'il me posait problème. Tout d'abord il me demandait beaucoup de temps. Comme la granularité du tableau de bord est censée être au niveau de lots de tâches, il me fallait hiérarchiser les tâches de base, puis les additionner, avant de les convertir en jours.

Ensuite, je n'avais pas l'expérience de l'environnement technique dans lequel j'évoluais, donc mes estimations étaient trop peu précises. Par exemple sur les tâches 1501 et 1504, l'avancement est bien supérieur au temps consommé. Pour la réalisation de l'aspect tangible, c'était encore plus flagrant, puisque j'ai consommé 15% du temps estimé pour arriver au but. Je n'étais pas en situation d'utiliser le jugement d'expert ou l'estimation par analogie. Et les méthodes d'estimation ascendante ou paramétrique me paraissent bien trop coûteuses en l'absence de besoin réel et motivé.

J'étais donc confronté à un tableau de bord dont j'avais beaucoup de mal à définir la granularité des tâches, dont les estimations n'étaient pas crédibles, et qui me demandait beaucoup de temps de mise à jour et de contrôle. J'ai plutôt eu le sentiment qu'il était adapté pour gérer un projet au forfait, donc avec un budget en jours-homme à répartir sur des lots de tâche puis à imputer chaque semaine. Mais dans mon cas, il ne m'apportait pas assez d'informations.

N'étant dans les faits pas contraint par des durées de tâches, et cherchant tout de même à être productif, j'ai décidé d'arrêter de maintenir ce tableau. Mais j'ai conservé l'idée de collecter toutes les données.

4.1.3 MÉTHODE ADOPTÉE

Ma posture a donc évolué, mais j'étais toujours dans l'optique de noter tous les temps pour ne surtout perdre aucune information. J'avais simplement dans l'idée que des synthèses offrant une vision d'ensemble pourraient toujours être générées a posteriori à partir du détail. Il m'a paru judicieux de catégoriser les temps consommés, en plus de les affecter à des tâches. Mon objectif a été d'une part de capitaliser le savoir pour analyser rétrospectivement ma propre activité, mais également d'offrir à mes tutrices et mes successeurs la possibilité de procéder à des estimations par analogie.

Tableau 6 : journal de suivi quotidien des temps consommés.

Les colonnes sont : la date du jour, les détails de l'activité, la catégorie (obligatoire), le domaine (optionnel), la durée en heures, le numéro de tâche, des commentaires (caché). Quelques couleurs me facilitent la lecture, par exemple les réalisations notables sont en vert.

Jour	Activité	Catégorie	Classement	Durée (/ 8h)	#
15/09/2015	Praxéologies : analyse du code source pour créer des fichiers JSON d'un format attendu	développement	Praxéologies	3	15037
15/09/2015	Mode auteur : Remplacer la valeur de la variable dans les{} : ne fonctionne pas dans la consigne d'une étape	développement	Auteur (module)	0,5	15034
15/09/2015	Mode auteur : Les valeurs des variables "previous_" ne sont pas remplacées dans les messages	développement	Auteur (module)	0,5	15035
16/09/2015	Mise en route de SimBuchettes sur Android	mise en route de l'existant	Dispositif externe	3	15038
16/09/2015	Réunion ELR - Cyrille Desmoulins pour le serveur Virtuoso qui ne fonctionne pas : rappel et état des lieux	réunion	Praxéologies	1	
16/09/2015	ANDROID : Début de création du code de notification de FormID	développement	Dispositif externe	4	15038
17/09/2015	FORMID : développement de l'adaptateur de communication avec SimBuchettes. La lecture de l'état est opérationnelle ! :-)	développement	Dispositif externe	3	15038
17/09/2015	Essai d'organisation des idées pour une version 5 du plan	rédaction du mémoire		2	
17/09/2015	Rendez-vous avec M. Fayolle poour faire le point sur le stage.	réunion		3	
18/09/2015	Les messages de succès d'exercice peuvent maintenant contenir des variables aussi	développement	Auteur (module)	2	15036
18/09/2015	Travail sur le problème des WebSockets que j'arrive en partie à reproduire. Mise à jour de "atmosphere" JS et Java	infrastructure	Elève (module)	6	
	SEMAINE 30				
21/09/2015	Praxéologies : avancement du fac-similé	développement	Praxéologies	2	15037
21/09/2015	Praxéologies : refonte du diagramme	rédaction du mémoire	Praxéologies	3	15037
21/09/2015	Point rapide avec Anne	réunion		0,5	
21/09/2015	WebSockets : début de migration vers des WebServices. Estimé : 2 semaines fini, stabilisé	développement	Elève (module)	2,5	15041
22/09/2015	WebSockets : suite de la migration pour s'en débarrasser. Améliorations sur la fin de l'exercice, sur l'affichage des messages dans la bonne zone, correction du fait que le message de succès de fin d'étape n'était pas visible pour la dernière, correction de "sauter l'étape" qui incrémentait de 1 même quand l'exercice était terminé	développement	Elève (module)	7	15041
22/09/2015	Rédaction du mémoire : introduction du LIG, MeTAH, FormID	rédaction du mémoire		1	
23/09/2015	Rédaction du mémoire : partie "Héritage"	rédaction du mémoire		8	

Le Tableau 6 montre un extrait de ce journal quotidien. Le total de chaque journée doit s'établir à huit heures. L'objectif est en effet de pouvoir convertir en jours-homme et de facilement contrôler la saisie des temps, plutôt que de mesurer les durées réelles. Ce journal est la base de l'intégralité de mon suivi.

Le journal des temps consommés au quotidien est secondé par une liste des tâches, illustrée Tableau 7.

Tableau 7 : liste des tâches en attente, en cours ou réalisées.

La tâche est numérotée manuellement. La colonne « état » contient la date de fin de réalisation si la tâche est terminée, ou une priorité dans les autres cas. Le total du temps passé dans la dernière colonne est calculé grâce au numéro de tâche du Tableau 6.

N° Tâche	Libellé	Etat	Temps passé
15024	Auteur : pouvoir réarmer un contrôle lorsque l'on se rapproche du but à atteindre (nb de lignes/colonnes du carré magique par exemple)	12/08/2015	6,0
	Dispositifs : trouver une solution quand plusieurs dispositifs sont connectés ou notifient Formid - rendre l'affichage des variables utile car pour l'instant illisible, et le cas échéant l'enlever - que le choix effectué ait un réel caractère filtrant auprès de Formid		
15025	- automatiser le déclenchement du choix s'il y a plusieurs dispositifs lors de l'initialisation du mode élève.	06/08/2015	24,0
15026	Auteur : que le bouton de choix du dispositif externe n'apparaisse que pour les cas où il y a plusieurs dispositifs externes... OU .... dans les cas où il peut y avoir plusieurs dispositifs externes (TPElec: non, SimuBuchettes:Oui)	06/08/2015	1,0
15027	Auteur : que le bouton de changement d'état n'apparaisse que si un type de dispositif externe est choisi	07/08/2015	1,0
15028	Modifier le serveur Formid pour permettre un mode élève déporté	12/08/2015	31,0
15029	Admin : l'ajout de beaucoup de disciplines dépasse en bas de l'écran	02/09/2015	0,5
15030	Choix du dispositif externe : le compteur de durée est faux si la date du serveur n'est pas la même que la date du client	01/09/2015	4,0

Les tâches sont ajoutées au fil de l'eau, même lorsqu'elles sont incertaines ou qu'elles concernent une vérification, afin de ne rien omettre. Lors de certaines phases un peu chargées, un point d'équipe donne lieu à une liste de priorités, qui sont reportées dans la colonne « Etat ». Je n'ai pas conservé l'historique des priorités puisque ces dernières n'ont de sens qu'au regard du lot de tâches duquel elles faisaient partie à un instant précis. Ma méthode de suivi m'a donc permis de consigner toutes les informations que j'ai jugées pertinentes.

4.1.4 PÉRENNITÉ DE FORMID

Il semble vraiment intéressant de se pencher sur l'évolution de l'informatique. Pour développer une application, il est nécessaire de choisir des technologies de développement. Ces choix semblent les meilleurs au moment où ils sont faits, et prennent en considération la probabilité de stabilité dans le temps.

Mais l'expérience montre que tout évolue. Par exemple les composants Java côté client (dans le navigateur internet) étaient vraiment prometteurs pendant des années, et FORMID a profité de cette possibilité. La première version de la déclaration des impôts sur le revenu en ligne utilisait aussi un composant Java côté client. L'expérience montre que ces composants disparaissent. La faute en partie aux politiques de sécurité toujours plus restrictives et aux mises à jour incessantes. Hors du monde du libre, Microsoft aussi est touché, il abandonne le développement de Silverlight, un composant côté client concurrent de Adobe Flash et qui se développe en langage ".NET".

Pendant ce temps, les composants logiciels évoluent. jQuery sort en version 2, AngularJS également. Ces technologies sont beaucoup utilisées par FORMID. Mettre à jour ces composants a un prix. Pour jQuery cela se fera au détriment des anciens navigateurs qui ne seront plus supportés, et pour AngularJS à priori au prix d'un redéveloppement complet. Maintenir les composants à jour est donc totalement irréaliste du point de vue du temps passé.

D'un point de vue général, il pourrait sembler avantageux de n'utiliser aucun composant logiciel tiers dans le développement d'un produit. Mais il serait utopique d'espérer développer les mêmes fonctionnalités que des cadriciels conçus par des grands groupes, et issus de plusieurs années d'évolution. De plus, même si les composants développés en interne deviennent puissants et fiables, pendant ce temps les navigateurs internet évoluent, et un jour le nouveau standard de JavaScript « EMACScript 6 » sera la norme, et il faudra redévelopper l'intégralité du code source, composants compris.

Je ne conçois donc pas qu'il soit possible de développer un logiciel dans des technologies qui ne nécessiteraient aucune réingénierie. L'obsolescence inévitable des composants doit donc être acceptée.

Il conviendra donc de surveiller l'évolution des technologies utilisées par FORMID pour décider d'opérations de remaniements techniques, comme cela a déjà été fait par le passé.

4.1.5 COMMUNICATION MOA – MOE

Du point de vue du produit FORMID, mesdames Guéraud et Lejeune représentent la maîtrise d'ouvrage (MOA). En effet, elles me communiquent leurs souhaits et exigences, je fais avec elles des réunions d'avancement, et ce sont elles qui jugent de la qualité de ce qui est produit par rapport à leurs attentes. Je me place donc en tant que maîtrise d'œuvre (MOE).

Avant de parler des organisations de travail envisageables pour le projet FORMID, il convient de rappeler le contexte. Mesdames Guéraud et Lejeune sont des enseignantes-chercheuses qui sont accaparées par de multiples tâches d'enseignement et d'administration de l'enseignement. Le temps de recherche s'articule donc autour de ces contraintes et se révèle être très variable d'une semaine à l'autre. Cela pour dire qu'il ne leur est matériellement pas possible de s'engager à dates fixes.

Il n'est donc pas envisageable de se voir à intervalles réguliers. Il n'est pas non plus garanti de pouvoir obtenir une réponse rapide concernant un doute ou un point de décision. Tout ceci ne pose pas le moindre problème, à condition de savoir adapter sa méthode de travail. En premier lieu, je profite de la moindre disponibilité pour faire le point toutes affaires cessantes. Ensuite, il est primordial de ne jamais avoir qu'une seule tâche en vue, pour pouvoir basculer d'un point à l'autre en cas de blocage.

Du point de vue de la communication, j'ai eu le feu vert de mes responsables pour envoyer autant de courriels et de documentations que nécessaire. Je ne me suis donc jamais restreint.

Dans les faits, je n'ai jamais été bloqué. Tout d'abord, mes responsables se sont montrées très réactives, et ensuite parce que je ne reste jamais en attente. Habitué à travailler en autonomie, j'estime être force de proposition. Voici donc la position que j'ai adoptée face à chaque type de situation.

4.1.5.1 Tâches planifiées

Tout d'abord, il est utile de rappeler que lorsque les tâches ont été le fruit d'un point projet avec les responsables il n'y a en général pas d'inconnues sur le travail à réaliser. Ces tâches sont habituelles.

4.1.5.2 Besoin de perception de l'objectif

Dans certains cas j'ai besoin de savoir comment le produit devrait être dans l'idéal. Ce genre de considérations de stratégie à long terme ne transparaît pas forcément à travers les autres fonctionnalités développées, c'est pourquoi il m'est arrivé plusieurs fois de questionner mes responsables. Lorsque la philosophie globale d'un logiciel est comprise, il est plus facile de prendre des initiatives qui ne risquent pas de diverger de l'objectif à long terme.

4.1.5.3 Améliorations courantes

En utilisant le logiciel, si je me rends compte que certaines opérations sont peu ergonomiques, répétitives, ou contre-intuitives, je prends l'initiative d'apporter des améliorations. Dans ce genre de cas, il arrive fréquemment que je réalise, puis que j'informe. Ce mécanisme de réalisation avant approbation est une méthode que j'ai déjà pratiquée par le passé en travaillant sur des développements en mode prototype. Beaucoup d'améliorations ont été réalisées de cette manière, notamment dans le module auteur, mais pas seulement.

Le réusinage de code (en général nommé *refactoring*), activité commune au quotidien également, ne fait jamais l'objet de mails d'information puisque l'impact final est complètement invisible par nature.

4.1.5.4 Évolutions significatives

Dans le cas d'évolutions qui ont un impact fort en termes de temps de développement, j'écris un document adapté qui explique la situation actuelle et la ou les solution(s) envisagée(s).

Dans le cas de l'évolution des dispositifs externes, le cas le plus important, le document décrit l'état actuel, l'état fonctionnel avec une évolution technique mineure, et l'état idéal avec une évolution technique majeure. Ce genre de tâches nécessite généralement de reparler de mon projet de vive voix. D'abord pour que je puisse expliquer la cause de ma proposition et l'avancée qui serait obtenue, mais aussi pour évoquer les éventuels inconvénients, les risques et le temps de développement estimé.

Dans ce genre de chantiers importants, j'aime sécuriser ma démarche par une évolution incrémentale.

4.1.5.5 Télétravail

Pendant mon stage, il m'a été offert la possibilité par mesdames Guéraud et Lejeune de travailler depuis mon domicile pendant la période de fermeture estivale des bureaux MeTAH. J'ai vraiment apprécié cette preuve de confiance, et j'ai voulu œuvrer pour m'en montrer digne.

J'avais des objectifs ambitieux pour cette période, établis en collaboration avec mes responsables. L'été était le moment idéal pour effectuer des modifications lourdes dans FORMID, puisque je ne pouvais pas être interrompu dans mon travail. Les tâches concernaient la communication avec les dispositifs externes et le mode passif de FORMID, ainsi que le mode élève déporté dans le dispositif externe.

Du fait du télétravail, j'ai consigné toutes les heures effectuées dans mon journal de suivi quotidien pour en avoir une bonne visibilité. Cela m'a permis d'équilibrer la durée du travail sur plusieurs jours. Comme je conservais toujours de l'avance pour m'autoriser un travail moins fourni le lendemain, j'ai terminé avec une moyenne de 22 minutes de supplément par jour.

Du point de vue de la communication, j'ai beaucoup informé mes responsables pour leur donner les moyens de me suivre :

- Chaque matin j'envoyais un courriel expliquant ce que j'avais prévu de faire dans la journée.
- Lors d'avancées notables, j'expliquais ce qui avait été réalisé.
- J'envoyais également les nouveaux documents rédigés.

D'un point de vue personnel, j'en ai déduit qu'il ne doit pas être aisé de piloter un travail à distance. Je pense qu'il est nécessaire d'accepter un certain niveau de confiance, et que le jugement doit concerner le résultat obtenu plus que les moyens mis en œuvre.

4.1.5.6 Synthèse

Je me suis parfaitement accommodé du caractère imprévisible de la disponibilité de mes responsables. La méthode de travail requiert certes beaucoup d'autonomie, mais j'ai beaucoup aimé avoir une importante marge de manœuvre pour proposer et réaliser mes idées. La motivation reste intacte, parce que j'ai toujours eu l'impression de produire un travail utile, notamment à travers la reconnaissance exprimée. L'encouragement produit par cette dynamique m'a incité à continuer à être force de proposition. Si je mentionne ces notions personnelles de motivation et de besoin d'accomplissement, c'est parce que je suis persuadé qu'elles sont la clé d'un travail en autonomie réussi.

4.1.6 CYCLE DE VIE DE FORMID

Le cycle de vie de FORMID est directement issu de son orientation pour la recherche. L'objectif du logiciel est de générer des traces qui doivent permettre de mieux comprendre l'utilisation et l'apprentissage, pour améliorer le produit. La Figure 53 illustre le fait que les évolutions de FORMID sont directement issues des travaux de recherche précédents, et préparent les suivants. Les traces analysées sont celles générées par les auteurs et les tuteurs. Les traces du travail des élèves quant à elles sont à destination des tuteurs et n'entrent pas en compte dans la conception suivant une expérimentation.

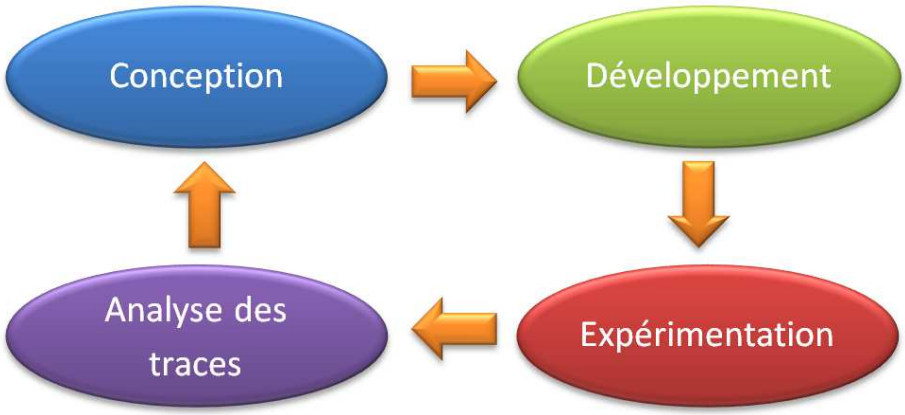


Figure 53 : cycle de vie simplifié de FORMID

Cette représentation est intéressante parce qu'elle illustre bien l'importance des traces générées pendant les expérimentations. Ce cycle présente une vision macroscopique de l'activité, puisqu'entre deux expérimentations nous concevons et réalisons beaucoup d'améliorations. Certaines phases de développement sont également parfois menées de manière prospective avec les utilisateurs, comme la conception et la réalisation des premières versions du module auteur.

4.1.7 MODÈLE DE DÉVELOPPEMENT

Un modèle de développement est un découpage temporel de ses différentes phases. Cette organisation permet de planifier les échéances et de cloisonner le travail en cours. Je vais présenter les modèles qui peuvent être utilisés avec FORMID et expliquer en quoi ils sont adaptés.

Citons pour placer le contexte le modèle le plus connu, dit « en cascade ». Dans ce modèle, toutes les phases se succèdent de manière linéaire : étude de faisabilité, définition des besoins, conception générale et ainsi de suite jusqu'à la phase de recette. Ce qui a été validé par le client sera développé en un bloc, puis livré. Mais les écueils sont multiples. Tout d'abord il est très difficile d'imaginer le résultat final sur la base de spécifications papier, aussi détaillées soient-elles. Mais comme le client ne pourra pas voir son produit entre la validation des spécifications et la livraison, ce modèle ne tolère aucune erreur ou imprécision. Or, ne serait-ce que dans la communication, les zones d'ombre, les erreurs d'interprétation et les imprécisions créent un écart entre ce qui est exprimé sur le papier et ce qui est compris.

La suite logicielle FORMID, issue de travaux de recherche, a une nature adaptative. C'est-à-dire que le futur s'écrit sur les bases de ce qui est approuvé aujourd'hui. Il n'est donc pas concevable d'appliquer un modèle en cascade. Mais les modèles suivants, flexibles, peuvent être tout à fait être adaptés selon le cas.

4.1.7.1 Le modèle du code-and-fix

Le modèle du code-and-fix (Figure 54) est dédié aux problématiques de faible ampleur dont le besoin est connu, mais pas la méthode de résolution. Cette approche de mise au point d'un développement par essais successifs est loin d'être une démarche rigoureuse et me semble donc peu sérieuse.

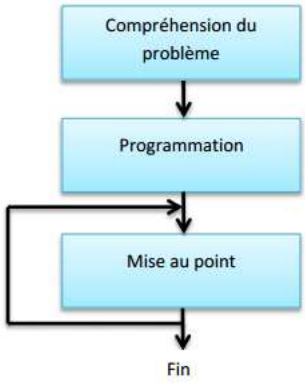


Figure 54 : le modèle de développement code-and-fix
Source : [**MORLEY08**], p40

Cependant, je reconnais utiliser, bien malgré moi, cette manière de procéder pour gérer les améliorations de la mise en page en HTML. L'objectif est rapidement identifié, voire spécifié par la MOA, mais la mise au point me demande toujours des dizaines d'ajustements successifs, comme expliqué dans la partie 2.2.1. Bien entendu, à part des phases de correction d'anomalies, il n'est pas souhaitable d'agir de cette manière.

4.1.7.2 Le développement itératif

Le développement itératif, et souvent incrémental (Figure 55), que l'on retrouve dans des méthodes agiles telles que SCRUM, consiste à atteindre un objectif par la réalisation d'une ébauche qui sera affinée à chaque nouveau cycle. C'est-à-dire que le client peut manipuler le produit de version *n* et ajuster les besoins exprimés. Cette méthode offre l'avantage de limiter considérablement les écarts entre le besoin du client et la réalisation finale. De plus, le client juge à chaque cycle s'il convient de continuer à améliorer chaque fonctionnalité.

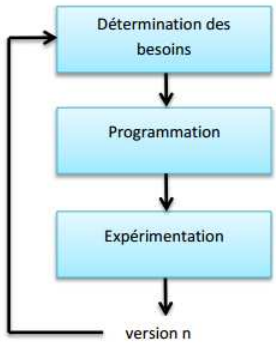


Figure 55 : le modèle de développement itératif
Source : [**MORLEY08**], p44

La manière dont j'ai réalisé une partie des améliorations du module auteur correspond tout à fait à un développement itératif. En effet, en utilisant le module auteur dans le cadre d'autres développements, comme la gestion des dispositifs externes, je me suis rendu compte de lourdeurs et de manques. J'ai donc amélioré de manière fréquente des éléments qui ne remettaient pas en cause la logique.

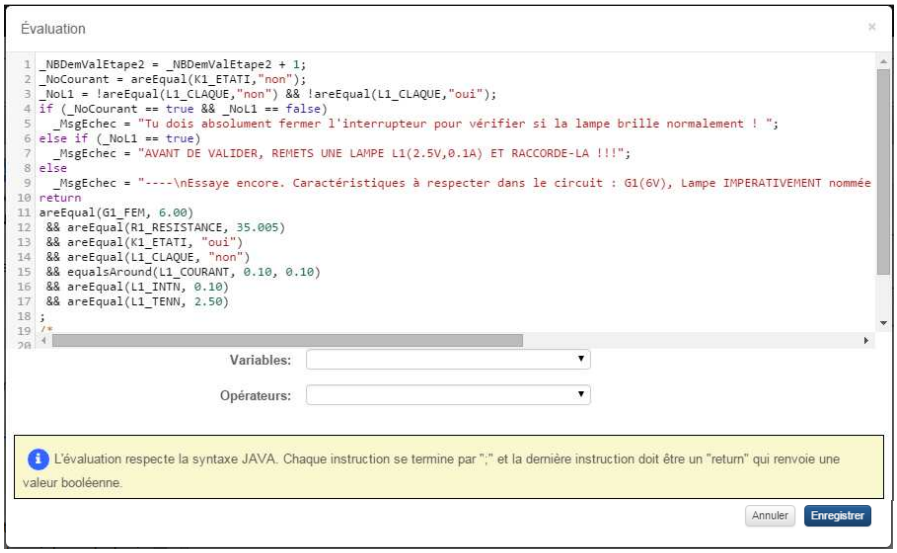


Figure 56 : écran de saisie de la condition d'évaluation d'un contrôle.

Comme exemple concret, l'écran de saisie du code Java représenté Figure 56 a fait l'objet d'évolutions successives pour aider l'auteur de scénario. Tout d'abord j'ai affiché une information qui indique qu'il s'agit de langage Java, puis par la suite j'ai ajouté la liste des variables existantes, suivi d'une liste d'opérateurs communs. Et enfin, bien plus tard, j'ai intégré le composant CodeMirror pour aider à la saisie du code Java, grâce à la coloration syntaxique, et à la mise en surbrillance de la correspondance des parenthèses et accolades.

4.1.7.3 Le modèle en W

Le modèle en W (Figure 57) permet une validation concrète, et même des expérimentations, grâce au développement de prototypes [**MORLEY08**]. Par rapport au modèle de développement itératif, nous nous trouvons dans des considérations conceptuelles qui permettent de concevoir des systèmes complexes et robustes. L'idée principale reste la même, à savoir que la manipulation du produit permet d'ajuster ou de définir les spécifications du cycle suivant.

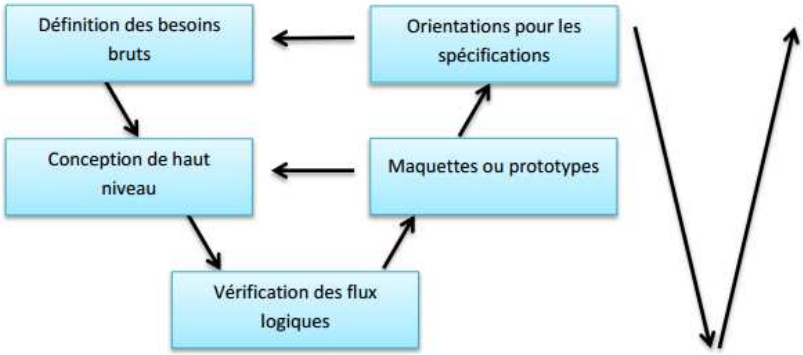


Figure 57 : le modèle de développement en W.
Source : [**MORLEY08**], p43

Cette mécanique semble adaptée à la description des évolutions majeures subies par FORMID, en ce qui concerne notamment les dispositifs externes. Le document de départ, une spécification que j'ai écrite et déjà mentionnée, a servi à réaliser plusieurs étapes décrites. Tout d'abord la simulation *LettersGame* a été développée, puis FORMID s'est vu doter de la capacité à écouter un dispositif externe, et enfin il est devenu capable de lui envoyer un état à afficher… et par la suite, d'autres idées non prévues ont été spécifiées puis réalisées dans la même logique. Tout le processus nécessitait réellement une réalisation par étapes. Tout d'abord, nous avions besoin de valider l'intérêt de cette évolution et l'acceptabilité du temps de développement. Et ensuite, il était difficile de penser en amont à tous les problèmes à résoudre. Il aurait certes été possible de tout spécifier, mais le volume de travail aurait paru trop important, puisque la première étape était déjà sérieuse.

La clé se situe au niveau de la certitude. Mes responsables et moi-même n'étions pas certains du bien-fondé de nos choix et de la possibilité de réaliser cette amélioration dans un temps raisonnable. Rappelons qu'à l'origine de ces développements, se situe uniquement l'ambition de pouvoir établir comme preuve de concept un lien entre FORMID et la simulation *SimuBûchettes* fonctionnant sur une tablette tactile. Nous ne pouvions pas valider un investissement conséquent dans ces conditions.

Porté par les succès et protégé par l'approche incrémentale, j'ai finalement amené mes réalisations au-delà du résultat initialement envisagé. En effet, le mode passif de FORMID et le mode élève déporté ont dépassé les étapes de preuve de concept et de prototype, pour devenir d'une part fonctionnels avec la simulation *LettersGame*, et d'autre part pour permettre la lecture de l'état de *SimuBûchettes* depuis FORMID.

4.1.7.4 Synthèse

Les différents modèles de développement sont chacun utiles dans des situations particulières. Leur essence commune est la nécessité de travailler de manière répétée sur des développements d'ampleur modérée. Ainsi, un résultat est très rapidement visible et permet des ajustements.

4.1.8 ANALYSE RÉTROSPECTIVE DE L'ACTIVITÉ

En synthèse, ma méthode de gestion conserve une trace réelle et précise de la vie du projet. Pour preuve, tous les temps consommés et toutes les dates d'événements qui apparaissent dans ce mémoire sont issus du journal détaillé précédemment. Celui-ci m'a également été utile à la construction de vues macroscopiques, comme la frise chronologique Figure 58, qui représente les événements notables pendant 9 mois. Ces derniers sont détaillés à l'Annexe 12.

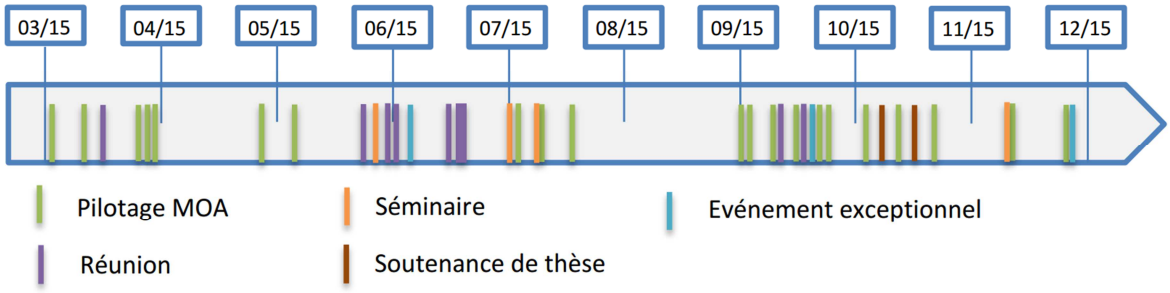


Figure 58 : frise chronologique des événements notables
La MOA est composée de mesdames Lejeune et Guéraud. Une réunion concerne au moins une personne étrangère à la MOA. Le premier événement exceptionnel est l'inauguration du bâtiment Amiqual4home le 05 juin, le deuxième la visite de M. Fayolle le 17 septembre, et le dernier l'expérimentation en classe de CE1 le 26 novembre.

Le Tableau 8 ci-dessous est rempli automatiquement à partir du journal d'activité qui a déjà été présenté (Cf. Tableau 6 page 74), et a permis la création de la Figure 59 représentant le temps consommé par type d'activité.

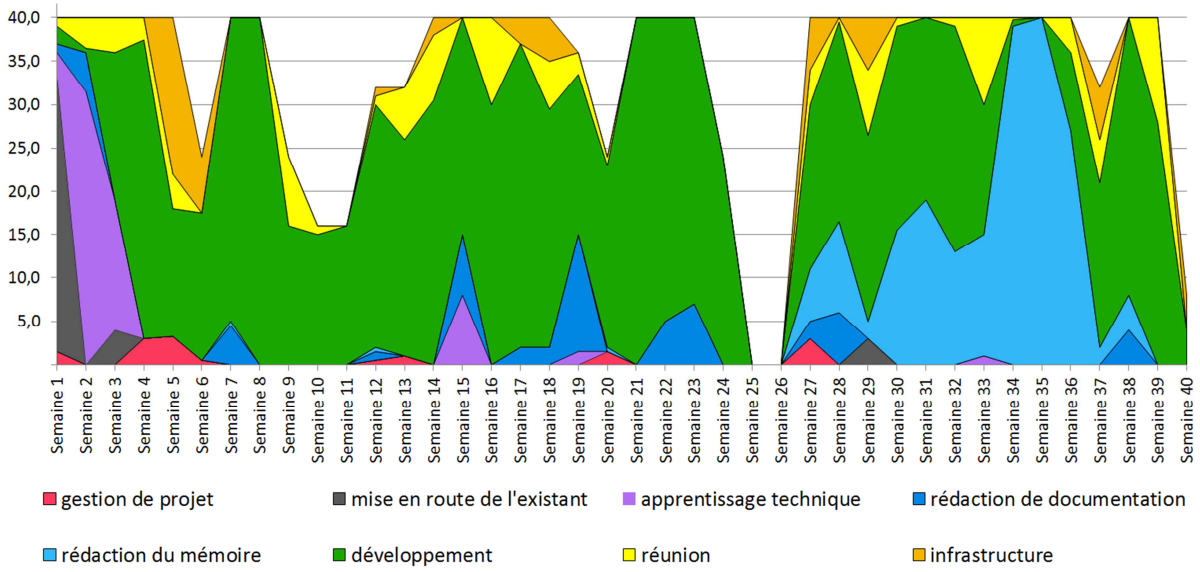


Figure 59 : consommation du temps par catégorie d'activité et par semaine.

Tableau 8 : temps consommé par catégorie d'activité et par semaine (extrait)

	Semaine 1	Semaine 2	Semaine 3	Semaine 4	Semaine 5	Semaine 6	Semaine 7	Semaine 8	Semaine 9	Semaine 10	Semaine 11	Semaine 12	S
	02/03/2015	09/03/2015	16/03/2015	23/03/2015	30/03/2015	06/04/2015	13/04/2015	20/04/2015	27/04/2015	04/05/2015	11/05/2015	18/05/2015	2
Catégorie	Heures	Heures	Heures	Heures	Heures	Heures	Heures	Heures	Heures	Heures	Heures	Heures	
mise en route de l'existant	32,0		4,0										
apprentissage technique	2,5	31,5	15,0										
gestion de projet	1,5			3,0	3,3	0,5						0,5	
rédaction de documentation	1,0	4,5					4,5					1,0	
rédaction du mémoire							0,5					0,5	
développement	2,0	0,5	17,0	34,5	14,8	17,0	35,0	40,0	16,0	15,0	16,0	28,0	
réunion	1,0	3,5	4,0	2,6	4,0				8,0	1,0		1,0	
infrastructure					18,0	6,5						1,0	
absence						16,0			16,0	24,0	24,0	8,0	
TOTAL	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	**40,0**	

Ces vues d'ensemble permettent une analyse visuelle très intéressante. Nous pouvons par exemple constater que les activités de la phase d'immersion (les 4 premières semaines) sont très différentes de celles qui suivent.

La Figure 60, accompagnée du Tableau 9, représente les chiffres totaux du temps consommé. La part du développement est naturellement très importante, mais la part prise par certaines autres activités peut être très instructive. La mise en route de l'existant regroupée avec l'infrastructure représente un problème qui pourrait sembler secondaire mais qui a demandé deux semaines-homme.

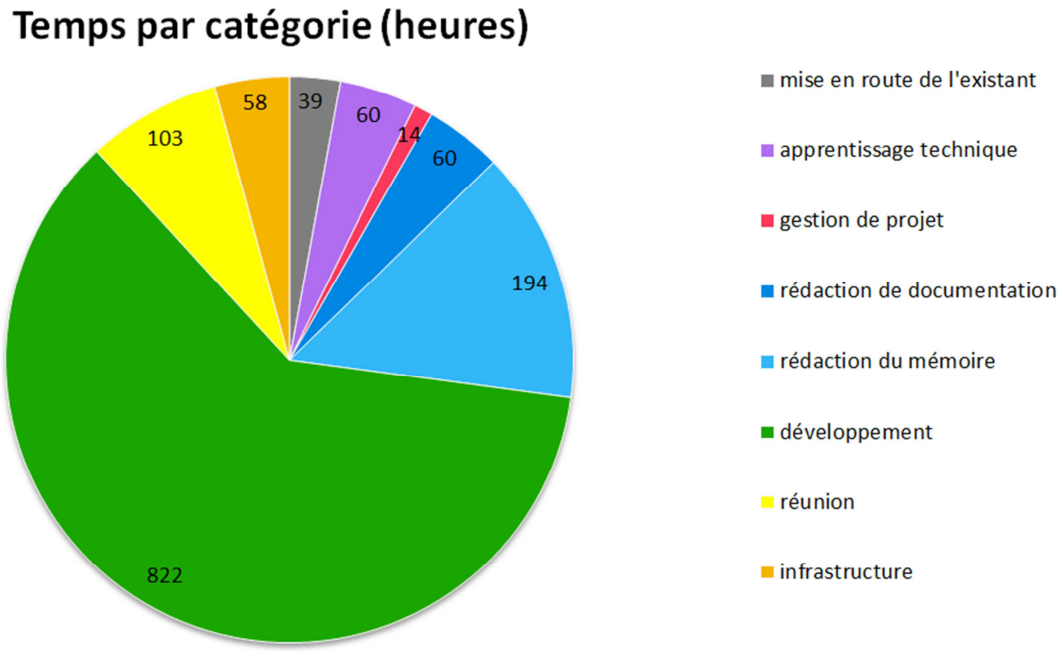


Figure 60 : consommation totale du temps par catégorie.

Tableau 9 : consommation totale du temps par catégorie.

Catégorie	Heures	%
mise en route de l'existant	39,0	2,9%
apprentissage technique	59,5	4,4%
gestion de projet	14,3	1,1%
rédaction de documentation	59,5	4,4%
rédaction du mémoire	193,5	14,4%
développement	822,0	61,0%
réunion	102,8	7,6%
infrastructure	57,5	4,3%
TOTAL	**1348**	**heures**
	168,50	**jours-homme**

Lors de mon arrivée, j'avais noté la prévision de la répartition de mon temps, pour pouvoir la confronter à la réalité. J'avais estimé que le développement occuperait 70% de mon temps, ce qui n'est pas loin de la réalité. Dans les chiffres notables, je pensais également réserver 5% de mon temps à la gestion de projet. L'expérience m'a montré que je n'avais pas de réels besoins de suivi et que dans ces conditions j'avais choisi de limiter le temps consommé.

Il me semble également intéressant d'entrer dans le détail et de faire ressortir quels ont été les temps consommés par module de FORMID (Cf. Figure 61). Je précise néanmoins que la vue est bien différente. Nous voyons des cumuls par domaine, toutes activités confondues. Par exemple le domaine "dispositif externe" regroupe du développement, de la rédaction de documentation et certaines réunions. Les 6 dernières colonnes reflètent simplement les temps du graphique précédent qui n'ont pas pu être ventilés dans les autres colonnes.

Les temps consommés les plus importants vont au module auteur à 15,7% et aux dispositifs externes à 17%.

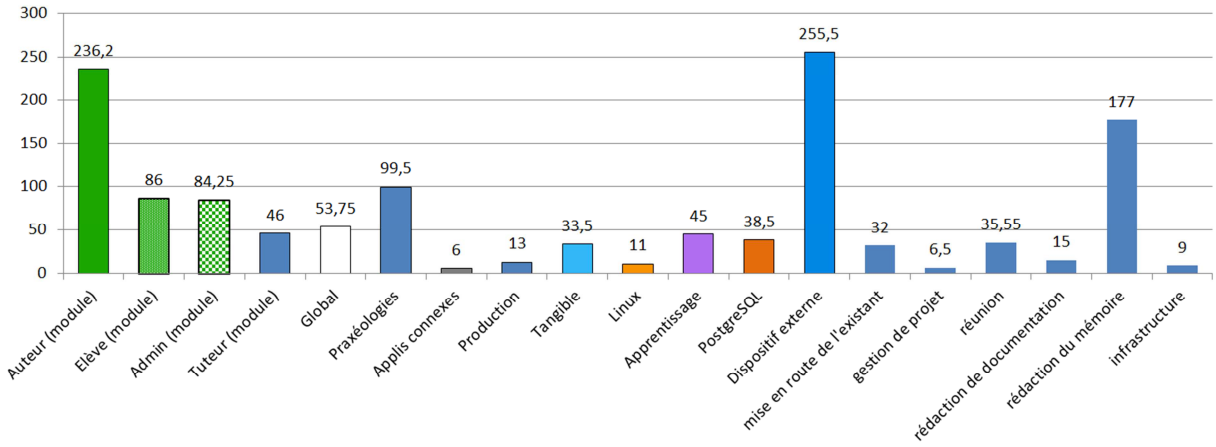


Figure 61 : nombre d'heures consommées par domaine

Je précise que le temps passé à la rédaction de documentation est sous-évalué, parce qu'il est parfois intégré dans d'autres activités. Par exemple un gros document a été créé au fur et à mesure de mon travail sur Linux et PostgreSQL. Une petite partie seulement de l'élaboration de ce document a été comptabilisée.

La rubrique "global" peut sembler étrange, ce sont les activités le plus souvent de développement qui concernent tout le logiciel, comme la gestion des sessions utilisateur, ou les messages avertissant que le travail sera perdu lorsque l'utilisateur essaye de quitter le module en cours.

En conclusion, toutes ces données sont le résultat de l'ensemble de mes heures consommées et représentent l'aboutissement de mon journal quotidien de suivi. J'ai bon espoir qu'elles permettent à mes responsables d'analyser mon stage et d'aider à prévoir le suivant.

4.1.9 SYNTHÈSE DE LA GESTION DE PROJET

Le développement du projet FORMID a été suivi de la manière qui m'a paru la plus fine, et cela m'a permis de sortir des indicateurs de consommation du temps qui donnent une vision globale du projet.

Je n'ai pas persévéré dans ma volonté de mettre en œuvre tous les aspects de la gestion de projet, comme l'estimation de charges, la planification et la gestion des risques. Tout d'abord parce que l'approche que j'en ai eue m'a donné un bon aperçu des contraintes, de la méthodologie à adopter et de mes besoins en terme de logiciel. Et ensuite, parce que cela m'aurait demandé énormément de temps, et que je ne voulais pas que cet objectif personnel interfère trop avec les enjeux principaux.

Je suis tout de même satisfait des résultats obtenus grâce à la journalisation d'informations qui se sont révélées pertinentes. Ils m'offrent une vision d'ensemble de ce qu'a été mon activité, ce qui me sera utile dans l'avenir. Ces données, d'ensemble comme de détail, peuvent également être capitalisées pour planifier le travail de mon successeur.

4.2 CERCLE RELATIONNEL

Pour mener à bien mes missions, j'ai été amené à communiquer avec de nombreux acteurs et actrices.

Si je souhaitais classer les types de relations, je pourrais mentionner celles directement liées à FORMID, aux dispositifs externes, au serveur d'ontologies et au dispositif tangible. Ensuite, je considérerais l'infrastructure, et je terminerais par toutes les autres relations qui m'ont permis de comprendre l'activité de MeTAH. En effet, ce n'est pas lors d'un seul séminaire, qui multiplie les termes techniques et ceux du domaine d'application, que j'ai pu comprendre ce qu'étaient les praxéologies. C'est la récurrence avec laquelle j'ai entendu les mêmes termes qui m'a permis de comprendre le lien entre tous les travaux apparemment sans rapport effectués ici.

Le Tableau 10 recense toutes les relations professionnelles que j'ai pu avoir, dans l'ordre de rencontre. L'objectif est de mettre en avant la richesse des personnes plus que de leur associer une étiquette. En effet, qu'il s'agisse de tangible, d'infrastructure ou de praxéologies, la finalité est toujours FORMID.

Tableau 10 : tissu relationnel, dans l'ordre chronologique de contact.

Prénom et nom	**Type de relation**
Anne Lejeune ***Maître de conférences***	Responsable *Formid*.
Viviane Guéraud ***Maître de conférences***	Responsable *Formid*.
Jacky Coutin ***Ingénieur Support (CNRS)***	J'ai eu besoin de ses services plusieurs fois pour la configuration du poste de travail, les autorisations d'accès au réseau local pour des périphériques comme le routeur Wi-Fi ou la tablette tactile.
Rémi Pincent ***Ingénieur (INRIA)***	Responsable du développement de la partie tangible, donc du mini PC et des Balances de pesée. C'est son cahier des charges de service REST que j'ai validé, avant de tester son service. Enfin il a configuré le réseau pour l'inauguration Amiqual4Home du 05 juin.
Jean-Louis Mas ***Ingénieur Support (CNRS)***	Je ne l'ai jamais rencontré physiquement mais il m'a fourni de très nombreuses explications quant au fonctionnement du serveur Linux de FORMID. Il a également testé à ma demande la possibilité d'embarquer le serveur dans une VM sur un PC portable pour faire des démonstrations en situation de configuration réseau inconnue.

Prénom et nom	**Type de relation**
Nathalie Brasset ***Doctorante***	Son travail se base en partie sur *SimuBûchettes* de **Pierre Tchounikine**. A mené des expérimentations en classe de *SimuBûchettes* avec **Nadine Mandran** et m'a fait, suite à cela, une démonstration et un retour. A suivi avec moi le transfert de compétences de **Jade Thiriat** concernant son application tactile *Bouchettes*. J'ai également assisté le 10 novembre 2015 à son séminaire "*Etude de facteurs liés à l'histoire didactique pour la conception d'un modèle de décision didactiques de l'enseignant dans un EIAH*".
Jade Thiriat ***Stagiaire de Vanda Luengo***	J'ai travaillé en collaboration avec elle pour partager le savoir et le matériel tangible pendant qu'elle développait son application sur tablette Android. J'ai pensé une configuration réseau adaptée à un usage partagé du dispositif tangible.
Nadine Mandran ***Ingénieur d'études (CNRS)***	Responsable de la partie conception de la plateforme Undertracks et en charge des expérimentions centrées utilisateurs au LIG. Son activité porte sur les méthodologies de production et d'analyse des données pour les expérimentations. Elle m'a donné des explications concernant UT, et la manière dont les données de Formid se présentaient. Elle a également mené des expérimentations en classe de *SimuBûchettes* avec **Nathalie Brasset** et m'a fait un retour. Responsable MeTAH du projet de bûchettes tangibles hors lien avec FORMID, elle a mené des expérimentations avec des enseignants et était présente à l'inauguration Amiqual4home.
Patrick Echterbille ***Précédent stagiaire Cnam sur FORMID***	Je l'ai contacté à deux reprises concernant la configuration du serveur Linux, pour lui demander son avis concernant le problème des WebSockets qui ne fonctionnaient plus.
Cyrille Desmoulins ***Maître de conférences***	Responsable du projet Ontoprax, l'éditeur de praxéologies stockées sous forme d'ontologies. J'ai assisté à son séminaire du 30 juin 2015 à ce propos. Il a contacté avec moi **Mohameth François** qui avait développé pour **Patrick Echterbille** un mécanisme permettant l'accès aux praxéologies depuis FORMID.
Gabrielle Devanz ***Stagiaire de Cyrille Desmoullins***	Elle a travaillé sur l'éditeur d'ontologies Ontoprax. Son poste de travail, ancien poste de **Mohameth François**, héberge les serveurs Apache et Virtuoso nécessaires à FORMID pour obtenir les praxéologies.
Patrick Wang ***Doctorant***	C'est grâce au routeur Wi-Fi qu'il a prêté à **Jade Thiriat** que celle-ci a pu travailler sur tablette tactile. Et c'est grâce à son prêt de routeur que j'ai pu réaliser une configuration réseau convenant à la fois à FORMID et à l'application tactile pour l'inauguration du 05 juin. Il m'a également fourni une tablette tactile pour mener à bien mes développements sur *SimuBûchettes*.
Hamid Chaachoua ***Maître de conférences***	Responsable du projet d'expérimentation en grandeur réelle avec mise à disposition de matériel dans les écoles, il est également une figure importante du projet T4TEL. Il a proposé que je sois l'unique interlocuteur de **Rémi Pincent** pour la partie tangible, pour clarifier une communication hasardeuse avec lui. Il a mené des réunions auxquelles j'ai participé, traitant notamment du plan directeur de déploiement des dispositifs tangibles et des tablettes tactiles. Il m'a demandé mon avis technique concernant l'infrastructure à mettre en place et les contraintes réseau.
Patricia Marzin ***Maître de conférences***	Elle m'a donné plusieurs explications concernant les praxéologies et leur terminologie.

Prénom et nom	**Type de relation**
Catherine Bonnat ***Doctorante***	J'ai assisté le 07 juillet 2015 à son séminaire exposant la didactique des levures entre la 6ème et la seconde. Les termes de praxéologie, de type de tâches et de technique, entendus pour la première fois ici, m'ont amené à terme à m'intéresser au sujet. Je lui ai demandé des explications concernant les praxéologies en général.
Pierre Tchounikine ***Professeur***	J'ai eu des échanges avec lui à propos du lien entre son application *SimuBûchettes* et FORMID. Il m'a exposé sa volonté de reconcevoir la simulation *SimuBûchettes* en intégrant les fonctionnalités dont j'avais besoin, et je lui ai fait part de mon désir de travailler tout d'abord sur la version courante pour preuve de concept.
Ben-Manson Toussaint ***Docteur depuis le 12 octobre 2015***	J'ai assisté à son séminaire concernant une modélisation des connaissances perceptivo-gestuelles, puis à sa soutenance de thèse le 12 octobre 2015. Les termes réentendus pendant son séminaire ont contribué à m'intéresser à la didactique et aux praxéologies.
Annie Bessot ***Collaborateur bénévole***	Je lui ai demandé de m'expliquer en quelques mots le concept de la théorie anthropologique du didactique (TAD) et le principe d'écologie. **Anne Lejeune** lui a demandé en ma présence des conseils relatifs aux praxéologies des mathématiques.
Jean-Michel Adam ***Maître de Conférences en informatique***	Responsable du micromonde *TPElec.* Je lui ai proposé de réaliser les modifications de son composant Flash pour améliorer FORMID.
Myriam Clouet ***Stagiaire M1 Informatique***	A développé la simulation *SimuBûchettes* sur Android sous la direction de **Pierre Tchounikine**.

Je souhaite par ce tableau démontrer que ne serait-ce que pour ce seul stage sur un produit créé par deux enseignantes-chercheuses, le tissu relationnel qui se créée devient vite conséquent. Les relations humaines sont donc primordiales pour obtenir toutes les informations du milieu dans lequel évolue FORMID. Ceci est dû à la nature pluridisciplinaire de l'équipe, et à la volonté de construire en collaboration avec les autres groupes de travail de l'équipe MeTAH.

4.3 BILAN

Voici donc l'heure du bilan. C'est une phase importante de création de valeur ajoutée qui ne doit pas être omise. Il est en effet difficile au quotidien d'avoir la conscience de tous les aspects de son activité. Je vais donc confronter l'état de développement aux objectifs initiaux, puis je listerai différentes idées d'évolution. Ensuite, j'analyserai mon travail et ce que j'en retiens.

4.3.1 ATTEINTE DES OBJECTIFS

Je vais exposer pour chacun des quatre points du sujet de mémoire d'ingénieur un court rappel, suivi d'une évaluation d'atteinte de l'objectif.

4.3.1.1 Évolution de la suite FORMID pour la prise en compte d'objets tangibles

Le dispositif tangible était déjà fonctionnel, mais le module de communication n'existait ni du côté FORMID ni du côté dispositif. La communication est maintenant établie, avec une infrastructure sous-jacente asynchrone. Cet objectif est donc complètement atteint.

Pour terminer de manière factuelle, j'aimerais présenter les étapes de réalisation du projet de connexion à un dispositif tangible :

Mars 2015 :

- **02/03/2015** *: Jour de mon arrivée.*
- **10/03/2015** : lecture et validation de la spécification du service REST, rédaction d'un document.
- **17/03/2015** : réunion avec Rémi Pincent, Anne Lejeune et Viviane Guéraud pour entériner la spécification et planifier la suite.
- **20/03/2015** : appel d'un service REST météorologique pour préparer l'appel du service REST tangible.

Mai 2015 :

- **18/05/2015** : mise à disposition du service REST par Rémi Pincent.
- **19/05/2015** : début de création de l'adaptateur de communication avec le dispositif tangible.
- **21/05/2015** : la communication fonctionne. Première démonstration à Anne Lejeune et Hamid Chaachoua. Il ne reste que très peu de temps avant la démonstration de début juin.

Juin 2015 :

- **05/06/2015** : Inauguration de l'Equipex Amiqual4Home. Je participe à la tenue d'un stand de démonstration (Cf. Annexe 2).

Novembre 2015 :

- **23/11/2015** : FORMID permet de gérer plusieurs dispositifs tangibles et d'en choisir un pour travailler.
- **24/11/2015** : réception des dispositifs tangibles.
- **26/11/2015** : expérimentation en classe de CE1 avec 11 élèves sur deux dispositifs tangibles.
- **30/11/2015 :** *Mon dernier jour de stage.*

FORMID est donc capable, suite à ces développements, de se connecter à tout service (web ou REST) asynchrone et de lire des données au format XML ou JSON.

4.3.1.2 Assistance à la conception de scénario : renforcement et nouvelles évolutions

Cet objectif consiste à faire évoluer le concept de preuve en place. Il s'agit de charger les données ontologiques, qui sont en rapport avec la discipline et le niveau en cours, afin d'aider l'auteur avec des données praxéologiques.

Nous n'avons pas pu établir de lien avec le serveur dédié. Nous avons donc créé des données artificielles simulant celui-ci, mais cette preuve de concept n'est pas ce qui était attendu. Finalement, nous avons développé notre propre serveur de praxéologies. Nous pouvons ainsi voir apparaître les données praxéologiques au sein du module auteur. L'objectif est donc très partiellement atteint, parce que le module auteur a évolué pour gérer le nouveau modèle praxéologique, mais il n'a pas pu faire l'objet d'expérimentations qui auraient permis de faire émerger de nouveaux besoins.

Étapes clé :

- **02/03/2015** *: Jour de mon arrivée. Le serveur d'ontologies est indisponible.*
- **19/06/2015 :** Le serveur d'ontologies est remis en route par Cyrille Desmoulins.
- **26/06/2015 :** Tentative de remise en route du serveur d'ontologies qui ne fonctionne plus.
- **09/09/2015 :** Début d'un fac-similé de praxéologies dans FORMID.
- **13/10/2015 :** Décision de renforcer le fac-similé avec des données praxéologiques réalistes.

Nous avons envisagé plusieurs possibilités. Le plus simple était bien sûr que le serveur d'ontologies soit réparé. La seconde solution consistait à reprendre les données praxéologiques de l'équipe Ontoprax, qui sont stockées dans un gros fichier XML, pour les intégrer dans une base de données qui remplacerait ce que faisait Virtuoso. Cependant, l'aspect « erreurs courantes » n'était pas géré, pas même au niveau conceptuel. Et enfin, la dernière solution était de créer un modèle de données au sein du groupe de travail FORMID et d'y saisir nos données praxéologiques. Nous avons opté pour cette dernière solution, en envisageant de développer plus tard une interface qui importerait automatiquement les praxéologies depuis le serveur de l'équipe Ontoprax :

- **09/11/2015 :** Développement de notre propre serveur de praxéologies (en PHP).
- **23/11/2015 :** Notre propre serveur de praxéologies est opérationnel, et FORMID amélioré en conséquence.
- **30/11/2015 :** *Mon dernier jour de stage.*

La situation a donc pu être débloquée in-extremis : nous sommes capables d'afficher des données praxéologiques au sein de FORMID dans le module auteur, d'associer un type de tâche à chaque étape et des erreurs courantes aux contrôles de situations observables.

4.3.1.3 Assistance du suivi : conception et développement de nouvelles fonctionnalités

Cet objectif devait se baser sur le serveur de praxéologies, à la suite des évolutions attendues pour le module auteur. Il n'a donc pas été possible de réaliser la moindre amélioration sur ce point.

Le suivi du mode tuteur n'a pas fait l'objet d'évolutions majeures. Nous pouvons simplement noter une correction en profondeur de la mise en page, ainsi que des corrections d'anomalies concernant un décalage des étapes et de petites améliorations visuelles.

Cet objectif dépendant du précédent n'a pas pu être mené à bien.

4.3.1.4 Amélioration, exploitation en vraie grandeur et documentation de la suite FORMID

Cet objectif, qui couvre la suite logicielle FORMID dans sa globalité, a fait l'objet d'énormément de travail cumulé. Les améliorations les plus importantes concernent la gestion des dispositifs externes et l'évolution du mode auteur, mais chaque élément a au moins été un peu enrichi.

1. **Module d'administration**

Le module d'administration permet dorénavant d'inscrire des exercices à des cours, et de gérer les séances de travail. Il permet également de définir les dispositifs externes liés à FORMID et leurs disciplines/niveaux scolaires associés, mais également leur configuration. En effet, celle-ci étant stockée dans différents fichiers sur le serveur Linux, elle était complètement inaccessible à un non-développeur.

2. **Module auteur**

Concernant le module auteur, il donnait le sentiment, à la fois dans le code et dans les fonctionnalités, que le gros œuvre avait été fait mais qu'il n'avait pas été finalisé. Par exemple la possibilité pour l'auteur de tester son scénario en mode exécution était à l'état embryonnaire, c'est-à-dire que le programme contenait la trame, mais j'ai dû résoudre huit erreurs successives, puis trouver une solution au problème conséquent de double-instanciation du composant Flash dans une même page. D'autre part il n'était par exemple pas possible de rouvrir un scénario publié. L'ensemble des fonctionnalités développées rendent le composant tout à fait fonctionnel sur ses différents aspects. Du travail a été fait pour guider l'auteur, sécuriser ses actions, et interdire certains comportements indésirables.

3. **Module élève**

La partie élève n'a subi que des améliorations de surface, à savoir une meilleure gestion de la mise en page, des boutons d'action affichés selon le contexte, bien visibles et colorés, et une synthèse vocale. Ce module était déjà bien terminé et n'a pas nécessité d'évolutions fonctionnelles notables. Nous pouvons rappeler la fiabilisation de l'utilisation du module par la suppression des WebSockets.

Le module élève déporté dans le dispositif externe fonctionne, mais il demande bien sûr que soit modifié le code du dispositif externe. Bien que la logique de FORMID soit de s'interfacer de manière générique, je pense que cette possibilité est très intéressante pour certains cas d'utilisation afin de se libérer de la contrainte d'avoir un ordinateur pour chaque élève, par exemple si le dispositif externe est sur une tablette tactile.

Le modèle élève est donc dans un état satisfaisant.

4. **Module tuteur**

Des informations se décalaient dans les colonnes, c'est-à-dire qu'un contrôle N°2 déclenché s'affichait dans la colonne N°1. Le module tuteur n'était pas en cause, ce sont les traces élève qui étaient mal enregistrées depuis les gros développements de 2014, d'où l'importance de corriger le problème.

Sinon le module lui-même a été retouché pour gérer la mise en page, car il était inutilisable en basse résolution. Un indicateur visuel de la progression du suivi avec une icône de battement cardiaque a également été ajouté.

Ce module est donc fonctionnel, et ce quelle que soit la résolution.

5. **Dispositifs externes**

Quant aux dispositifs externes, l'expansion rapide de leur maîtrise et de leur liaison a déjà été décrite en détail précédemment. Nous avons obtenu une version bien plus avancée qu'initialement prévu. Ce résultat est très satisfaisant puisque FORMID est une plateforme pédagogique qui se base sur des dispositifs externes : plus leur nombre est important et les capacités de liaison étendues, plus le discours décrivant FORMID est justifié. FORMID avait également à tort l'étiquette d'être dédié à l'électricité. La variété des dispositifs externes dorénavant gérés aura sans aucun doute une influence très positive sur l'image de ce logiciel. Enfin, la preuve de concept de la liaison à la simulation sur tablette tactile *SimuBûchettes* me paraît vraiment intéressante pour montrer que FORMID peut être présent sur tous les terrains. C'est tout du moins l'image qu'il doit véhiculer.

6. **Documentation**

J'ai créé de très nombreuses documentations techniques (Cf. partie 3.4.4). Je n'ai par contre pas pris l'initiative de réaliser de documentations utilisateur, et ce pour plusieurs raisons. Tout d'abord au début de mon stage Anne Lejeune m'avait expliqué que la connaissance fonctionnelle ne leur posait aucun problème. D'autre part, il existe dans l'histoire de FORMID plusieurs documentations utilisateur conséquentes, et en réécrire une sous-entend d'intégrer les bonnes idées des autres, tout en gardant une cohérence d'ensemble. Cela me semblait chronophage et annexe à mes objectifs. J'ai donc communiqué chaque amélioration fonctionnelle mais je n'ai pas rédigé de documentation utilisateur. Par contre il m'a été fait part du souhait que je documente beaucoup techniquement, ce qui pour moi s'inscrit tout à fait logiquement dans mon travail quotidien.

7. **Environnement de développement**

Le développement a été immensément facilité avec la migration de la base de données de H2 vers PostgreSQL, ainsi qu'avec la possibilité de déployer directement depuis l'environnement de développement NetBeans vers le serveur applicatif Wildfly.

Le quotidien du développeur a donc été amélioré.

8. **Généralités**

Le maintien en vie de la session d'un utilisateur ne fonctionnait pas, donc chacun perdait sa session quoi qu'il arrive après le délai paramétré. Le mécanisme en place ne gérait aussi que les échanges client-serveur, donc lorsque l'auteur travaille sur un scénario pendant longtemps sans contacter le serveur, il perdait son scénario sans même avoir la possibilité de le sauvegarder en local. J'ai mis en place un système général de renouvellement du jeton de sécurité, par des requêtes invisibles pour l'utilisateur, quel que soit l'écran sur lequel il travaille. La gestion des erreurs a également été améliorée pour afficher un message, et le cas échéant rediriger vers la page appropriée.

La gestion des sessions est vraiment importante car nous étions déconnectés très fréquemment. Quant à la gestion des erreurs, je constate que l'affichage ne fonctionne que dans des cas bien particuliers, et que le reste du temps il faut consulter la console du navigateur ou le fichier du serveur FORMID. Il reste encore une marge de progression à ce niveau.

4.3.1.5 Synthèse des objectifs

L'analyse des objectifs a permis de montrer que certains points importants du sujet n'ont pas pu être traités à cause de facteurs extérieurs. Ce temps prévu a été utilisé à bon escient, pour améliorer FORMID sur d'autres aspects. Bien que les avancées soient notables, un point demeure prioritaire. Les praxéologies sont en effet indispensables à l'avenir de FORMID, or elles n'ont pu être débloquées que très tardivement. Elles devront donc faire l'objet d'une attention particulière dans les modules auteur et tuteur.

4.3.2 PROPOSITION D'ÉVOLUTIONS

Après avoir longuement erré dans le code et les fonctionnalités de FORMID pour diverses raisons, je suis maintenant en mesure de proposer les évolutions qui me paraîtraient intéressantes. Certaines me semblent incontournables, d'autres beaucoup plus discutables.

4.3.2.1 Communication entre FORMID et le dispositif tangible

FORMID contacte à intervalles très réguliers le service du dispositif tangible, pour obtenir son état. Il serait maintenant possible, et bien plus logique, que FORMID soit en écoute (mode passif) et que le dispositif tangible le notifie de ses changements d'état. Cela demanderait une évolution du côté du dispositif externe, pour qu'il envoie chaque nouvel état vers le service web dédié, à l'adresse (pour le dispositif d'identifiant « 1 ») :

```
http://formid.imag.fr/ExternalDeviceNotification?devicename=BUCHETTES&command=setState&id=1
```

Ensuite du côté FORMID, l'adaptateur de communication devrait être modifié pour agir comme pour *SimuBûchettes*, c'est-à-dire demander le dernier état connu du dispositif tangible au serveur FORMID.

4.3.2.2 Module d'administration

Le module d'administration ne permet pas encore de saisir toutes les données. Il n'est en effet pas possible de saisir les rôles des utilisateurs, les pays, les niveaux scolaires, pas plus que les disciplines par niveau scolaire. Il faudrait également prévoir l'export des traces auteur, mais plutôt à terme car les données consignées sont pour l'heure embryonnaires.

4.3.2.3 Présence de multiples dispositifs tangibles

La montée en puissance des dispositifs tangibles a logiquement mené à la présence de plusieurs groupes de balances de pesée, pour différents élèves simultanément. Jusqu'à la dernière semaine de mon intervention nous n'avions qu'un seul dispositif avec deux balances, destinées au même apprenant.

J'avais posé la question à l'ingénieur en charge de savoir s'il y aurait ou non un mini-PC par élève. Sa volonté était au contraire de n'avoir qu'un seul mini-PC qui gère l'ensemble des balances de pesée. Cette solution simplifierait la communication, avec un seul interlocuteur pour FORMID. Cependant, il apparaît, d'après mes tests, que la connexion Bluetooth ne fonctionne que jusqu'à un mètre environ. L'ingénieur m'a également confié que la connexion supportait déjà difficilement quatre périphériques.

Finalement, un mini-PC accompagne chaque lot de balances. La solution rapide retenue pour permettre l'expérimentation en classe n'est peut-être pas la plus optimale, car c'est FORMID qui interroge le dispositif approprié. J'ai imaginé de faire choisir à l'utilisateur parmi plusieurs lignes paramétrées comme s'il s'agissait d'exemplaires détectés de dispositifs externes.

Il me semble logique de devoir envisager pour chaque mini-PC qu'il soit à l'initiative du contact avec FORMID. Avec ce fonctionnement déjà développé pour *SimuBûchettes*, « FORMID serveur » en mode passif écoute et conserve le dernier état reçu de chaque dispositif externe. Chacun doit avoir un identifiant unique alphanumérique. Cette évolution nécessite côté FORMID que soit simplement modifiée l'interface de communication pour qu'elle fonctionne comme celle de *SimuBûchettes*. Cela devra faire l'objet d'un cahier des charges pour le développement du côté mini-PC.

Pour synthétiser, FORMID a pu être adapté rapidement pour gérer de multiples dispositifs tangibles, et peut être encore amélioré avec peu de développement.

4.3.2.4 Module auteur

Le module auteur pourrait être amélioré pour tracer toutes les actions importantes. Celles-ci doivent préalablement être définies avec soin.

La recherche d'un exercice publié pourrait également être facilitée en offrant la possibilité de filtrer par type de dispositif externe, par auteur ou par nom.

Malgré l'amélioration de l'ergonomie, l'auteur aura bien du mal à comprendre seul la logique complète de ce module, c'est pourquoi il serait à mon avis bénéfique d'ajouter une aide sur l'écran. L'idéal étant de la diviser en petites informations disséminées. Cela pourrait être à travers la création tutorée d'un scénario pédagogique prédéfini.

Dans le module auteur, un assistant permet de créer un contrôle de situation observable, sans avoir à taper de code Java. Par contre il n'est pas possible de modifier ce contrôle, autrement qu'en intervenant dans le code. Il serait bien de pouvoir rouvrir l'assistant et que celui-ci, après avoir interprété le code Java, sélectionne à nouveau les choix qui avaient été faits. Je propose, pour éviter d'avoir à développer de difficiles interprétations, que l'assistant de création génère, en plus du code, un commentaire qui soit lisible par le programme.

Viviane Guéraud avait évoqué avec moi l'idée que l'auteur puisse créer ses variables calculées. L'idée me semble intéressante, et en voilà mon analyse. Cela demande tout d'abord d'uniformiser les codes qui convertissent les états en variables. La plupart sont en effet côté client, au sein de chaque adaptateur de communication, sauf ceux de *SimuBûchettes* et de la version externe de *LettersGame*. Il faudrait que tout passe côté serveur, donc transformer le code JavaScript en code Java, qui est un langage compilé. C'est-à-dire que le calcul de la variable, s'il est saisi en Java par l'auteur, doit être compilé à la manière d'un scénario, l'idéal étant qu'il le soit à l'intérieur même du scénario. De plus, l'état doit transiter depuis le client vers le serveur lorsque c'est le client qui connaît l'état du dispositif externe. C'est donc une évolution délicate, à préparer avec un document de spécification et un diagramme de séquence détaillé.

4.3.2.5 Module élève

À l'heure actuelle, une zone est dédiée aux consignes, et une autre aux messages de rétroaction. Je pense qu'il serait plus lisible de présenter l'ensemble comme un journal. C'est-à-dire présenter tous les messages de manière chronologique, avec des styles d'affichage bien différenciés. Et pour éviter qu'une suite d'erreurs ne fasse disparaître la consigne, il serait peut-être intéressant de laisser visible la dernière consigne. En tout état de cause, il me paraît important de travailler à la partie affichage puisqu'il est délicat de déterminer quelle information de la zone de consigne correspond à quelle information de la zone de rétroaction.

4.3.2.6 Module tuteur

Le module tuteur fonctionne bien et offre trois vues de granularités différentes. Cependant, je pense que la navigation entre les vues pourrait être rendue plus intuitive. À l'heure actuelle, la navigation s'effectue par des actions non évidentes : clic droit de la souris sur un élève, sélection d'une étape qui ne semble être qu'un en-tête de colonne... En jouant sur les couleurs également, déterminer à quel niveau nous nous situons pourrait devenir plus évident.

Il me paraîtrait également intéressant de pouvoir rejouer une séance à la vitesse réelle, ou accélérée. En effet, en rejouant une séance passée, les observations préenregistrées des élèves s'affichent à intervalles réguliers à l'écran, alors que certaines actions peuvent être espacées d'une minute et d'autres de quelques secondes. Il me paraîtrait donc intéressant de développer une boîte de contrôle comme celle d'une vidéo, c'est à dire avec une barre d'avancement interactive, et la possibilité d'avancer, reculer, mettre en pause, et qui mette en avant

l'aspect "temps" et plus seulement chronologie. Cette évolution devrait dans l'idéal s'accompagner d'une modification du système de requête au serveur, pour lire les actions par lots au lieu d'une par une, notamment pour le cas où plusieurs actions se déroulent dans un laps de temps plus court que le délai entre deux requêtes. L'information visuelle apportée par un respect du temps me paraît intéressante pour savoir par exemple si un élève a tenté plusieurs validations de suite dans la même minute ou s'il s'est à chaque fois donné le temps de la réflexion entre deux validations. Un élève qui échoue malgré des efforts de concentration devrait peut-être bénéficier d'une aide appropriée.

J'ai également une proposition de développement vraiment conséquent. A l'heure actuelle, le tuteur ne voit de chaque élève qu'une liste de variables par action. Il me paraîtrait vraiment intéressant de développer pour chaque dispositif externe une sorte de petit écran de télévision, qui permette de représenter ces valeurs de manière schématique. Ainsi le tuteur pourrait voir les plateaux du jeu de lettres, le contenu des boîtes tangibles, ou une liste de composants électriques avec leur état : dessiner une ampoule grillée ou non, une résistance avec sa valeur à côté, par exemple. Le tuteur pourrait ainsi sur l'écran de suivi d'un seul élève voir une représentation de son dispositif externe, comme s'il le voyait vraiment travailler. Cette représentation serait bien sûr spécifique comme l'est l'adaptateur de communication de chaque dispositif externe.

À la demande d'Anne Lejeune, il serait bien plus facile de comprendre une situation observable si les variables évaluées étaient mises en surbrillance dans l'infobulle. En effet, si le dispositif externe comprend vingt variables et qu'une seule est évaluée, le tuteur ne sait pas laquelle parmi les vingt a déclenché le contrôle. Ce développement demande par exemple de rechercher au moment de la création de l'infobulle si le code Java contient chacune des variables. Le tuteur peut uniquement à l'heure actuelle se baser sur le nom du contrôle.

4.3.2.7 Dispositifs externes

Il me parait intéressant de faire évoluer le jeu de lettres *LettersGame* pour augmenter ses capacités dans le but d'autoriser la conception de nouveaux types d'exercices.

Il me paraîtrait judicieux de chercher à créer un autre dispositif externe en rapport avec les mathématiques, à un niveau supérieur à la numération. En effet, l'équipe MeTAH dispose de didacticiens des mathématiques, et la résolution d'équations (du premier ou second degré) est souvent prise en exemple.

Enfin, je conseille de créer un jeu. C'était mon premier objectif lorsque j'ai conçu le dispositif externe *LettersGame*, mais je n'ai pas trouvé d'idée de jeu qui puisse être guidé par un scénario pédagogique avec des étapes de validation. Il n'y a rien de mieux que l'aspect ludique pour capter l'attention et pouvoir ainsi communiquer sur des sujets sérieux.

Concernant le dispositif tangible, ce que l'élève voit à l'écran est seulement la liste des boîtes détectées. Je pense qu'il serait intéressant de représenter schématiquement le nombre de bûchettes et de fagots mesurés dans chaque boîte. Bien sûr sans afficher les nombres de bûchettes mesurés, car cela pourrait donner la solution de l'exercice à l'élève. Ce schéma pourrait être utilisé en début d'exercice pour montrer à l'élève l'état initial attendu, par exemple en haut, et l'état actuel en bas. Sans même avoir à compter l'élève pourrait manipuler pour placer le dispositif tangible dans l'état attendu.

Enfin, concernant le micromonde électrique *TPElec*, il serait intéressant de pouvoir lui fournir le circuit au format XML directement par une commande JavaScript. Il ne gère à l'heure actuelle qu'une URL, ce qui contraint FORMID à envoyer le fichier XML dans un fichier temporaire sur le serveur. Cette évolution est mineure pour *TPElec*, mais elle apporterait beaucoup de souplesse car cela nous éviterait d'avoir à supprimer le composant de la page web avant de le recréer pour pouvoir changer son état.

4.3.2.8 Environnement de développement

Il serait intéressant de chercher à documenter l'aspect lancement de l'application FORMID sur le poste de développement. En effet, le lancement en local crée un fichier d'archive web qui est immédiatement déployé. Cela fonctionne comme sur un serveur, mais ce processus est extrêmement lent. Il serait intéressant de chercher s'il est possible de lancer le code source sans le déployer, pour gagner un temps précieux.

Concernant le nom de l'archive web, sa version est "formid 0.2", il serait intéressant de chercher comment la version est censée être modifiée, puisque qu'elle est indiquée en dur dans nombre de lignes de code qui désignent nommément l'archive web.

4.3.3 ANALYSE DE MON INTERVENTION

4.3.3.1 Remise en cause de l'existant

D'un point de vue global à l'application, j'ai supprimé ou profondément modifié plusieurs parties importantes alors en place.

Tout d'abord la base H2. Elle avait été mise en place par souci de simplicité d'installation, donc d'économie de temps pendant une phase d'importants travaux sur FORMID. C'est moi qui ai eu le temps de la faire migrer sur un gestionnaire plus robuste. Le choix initial n'est donc pas en cause, puisqu'un choix s'effectue au regard des ressources disponibles avec les informations à notre disposition.

Ensuite les WebSockets. Cette technologie est vraiment prometteuse. Mais ses avantages par rapport à des services web ne font pas encore l'objet d'un besoin de FORMID. Comme je n'ai pas réussi à maîtriser la source d'erreurs en un temps raisonnable, j'ai pris la décision de les supprimer. C'est un choix pragmatique, qui sera peut-être remis en cause par la suite. Dans l'immédiat, nous avons une solution fonctionnelle et fiable.

Finalement, la gestion des circuits de *TPElec* n'utilise plus le circuit du niveau "exercice" mais a intégré l'état à l'intérieur de chaque étape du scénario. Comme nous l'avons vu, c'est la gestion des circuits de *TPElec* qui a beaucoup limité FORMID par le passé. Il ne s'agissait pas d'un choix délibéré, mais de contraintes spécifiques qui n'avaient pas encore pu être levées.

Citons également le cas des adaptateurs de communication avec les dispositifs externes, qui sont passés d'un extrême à l'autre lorsque j'ai remplacé les fonctions JavaScript par des contrôleurs AngularJS. Les méthodes globales alors en place répondaient au besoin de communiquer avec une instance de *TPElec* ou de *TPNum* de manière synchrone. L'évolution de la communication a donc induit l'évolution des adaptateurs. Là non plus il ne s'agit pas d'une remise en cause d'un choix précédent.

4.3.3.2 Enseignements tirés de la gestion de projet

Méthode de gestion

Je n'avais pas pris conscience avant d'essayer de gérer le projet que l'outil choisi devait être déterminé par le besoin. J'avais en effet le sentiment qu'un projet se gérait toujours de la même manière. J'ai constaté par exemple que gérer un diagramme de Gantt ne se justifiait que si nous avions de réels besoins de gérer les dépendances entre lots de tâches. De la même manière, ce que j'ai compris du tableau de bord du chef de projet que j'ai mis en place, c'est qu'il sert à gérer le temps restant de développement. Donc dans un projet en mode recherche, prototype ou simplement Agile, je crois qu'il n'aurait pas vraiment de sens. Je veillerai donc à bien identifier les aspects que je souhaite garder sous maîtrise avant de choisir un outil. Je prends conscience également que la gestion de projet peut demander beaucoup de temps, par exemple si l'on souhaite gérer à la

fois le chemin critique entre les lots de tâches et l'avancement dans un tableau de bord. Finalement, je pense que si cela répond à mes besoins, utiliser de simples tableurs peut ne pas être si ridicule. Bien entendu il serait souhaitable d'utiliser un logiciel adapté pour gérer une équipe pour éviter d'avoir à consolider les données manuellement.

Comptage des heures

Le sujet des heures consommées me semble un point à approfondir. Comme un chef de projet fonctionne à l'unité de jour-homme, il a tout intérêt à ce que la somme des journées des membres de son équipe donne toujours 8 heures. En effet, si une personne fait des heures supplémentaires et travaille 12 heures sur une journée, le chef de projet risquerait d'imputer un jour et demi au projet alors qu'une seule journée physique s'est écoulée. Le même problème peut se poser de manière opposée avec une personne à temps partiel. J'ai utilisé ce comptage pour mon journal : chaque journée totalise 8 heures, alors que dans les faits j'en travaille 7.

Le comptage sur 8 heures facilite la conversion en jours-homme, mais par contre nous perdons une information précieuse. En effet, même si la connaissance globale de l'avancement est prioritaire, chaque projet doit permettre de capitaliser le savoir pour la suite. Or, si une équipe entière a effectué beaucoup d'heures supplémentaires sur un lot particulièrement technique, le temps comptabilisé pour réaliser ce genre de tâches sera largement sous-estimé.

Une idée pourrait être de saisir le temps réellement consommé, puis que le système calcule un ratio de journée pour les imputations du chef de projet. C'est-à-dire que le total d'une journée de présence physique fasse toujours 1. Cette solution me parait viable mais elle semble avoir le défaut de compliquer les tâches de vérification, par exemple pour contrôler la source de 2 journées-homme.

4.3.3.3 Valeur ajoutée par mon travail

Les améliorations apportées à la suite logicielle FORMID ont été expliquées dans les parties précédentes, et l'atteinte des objectifs a été précédemment évaluée. Je voudrais en plus mettre en lumière les avancées notables ou hors cadre.

Il est maintenant possible de connecter FORMID à un nouveau dispositif externe très facilement. Il suffit de créer un adaptateur de communication, qui implémente tout ou partie des méthodes du service de pilotage `SimulationProvider`, lequel sera son unique interlocuteur. L'étendue de l'implémentation dépend du type de dispositif externe (simulation ou micromonde), de la possibilité de définir son état, et de l'éventualité de sa présence en de multiples exemplaires. Il ne sera jamais nécessaire de toucher à une autre partie du code de FORMID. Le dispositif externe peut être à communication synchrone ou asynchrone, être interrogé ou notifier FORMID. Le processus de gestion des dispositifs externes multiples n'a besoin d'aucun ajustement pour proposer à l'utilisateur les différents exemplaires qui se sont fait connaître. Le mode élève déporté est documenté, c'est-à-dire qu'il est expliqué comment adapter un dispositif externe, ou un logiciel qui l'héberge, pour qu'il soit capable d'exécuter des exercices directement.

La création d'un serveur de praxéologies a permis de lever la contrainte de dépendance de FORMID. De nouveaux développements peuvent donc être envisagés, notamment les objectifs que je n'ai pas eu le loisir de réaliser. Ce serveur ultra léger, développé dans le langage très commun PHP, peut au besoin être délégué à un autre groupe de travail ou faire l'objet d'une intervention ciblée. En considérant par exemple que le serveur d'ontologies dont nous dépendions devienne à nouveau disponible, il serait intéressant que ce ne soit pas FORMID qui l'interroge directement, mais que notre serveur de praxéologies s'en nourrisse. Cela permettrait de traduire les données pour que le code principal de FORMID n'ait pas à être modifié.

Tous les aspects techniques rencontrés ont été documentés, et les choix argumentés. Cette transmission d'informations permettra de futures réflexions dans FORMID sans qu'on ait à présumer de mes intentions. Les décisions futures bénéficieront donc de mon éclairage, que ce soit pour poursuivre dans une voie que j'ai initiée ou au contraire pour la remettre en cause.

4.3.3.4 Accomplissement personnel

Tout d'abord, j'étais très motivé pour découvrir le monde de la recherche. Je tenais à savoir comment le quotidien se déroulait, quelles étaient les contraintes, les financements, et les problématiques des personnels en poste. Dans les sphères professionnelles où j'évolue, je ne côtoie pas d'enseignants-chercheurs (maîtres de conférences), pas plus que de doctorants. J'ai également eu l'opportunité de discuter avec des ingénieurs évoluant dans le monde de la recherche. Pour toutes ces raisons, j'ai vraiment apprécié cette immersion. L'accueil qui m'a été fait malgré mon statut de stagiaire a toujours été cordial et professionnel, j'ai même été agréablement surpris d'être considéré par tous comme un ingénieur.

Mon travail ne revêtant pas un caractère particulièrement critique ou urgent, je ne me suis pas autorisé à presser les différents acteurs, même lorsque j'étais en attente de nombreuses semaines. Ma communication consistait à demander des informations et à suggérer des actions. À partir du moment où tout était dit et après avoir reporté le dialogue à mes deux responsables, je ne relançais pas. Je n'y ai pas été incité, et il est difficile de juger de l'importance que revêt mon projet aux yeux des autres.

Voici ma vision du travail que j'ai pu effectuer. Le mode auteur a beaucoup évolué, notamment grâce à la possibilité de tester le scénario en cours d'élaboration, mais je constate que je n'ai pas innové. De la même façon, et malgré d'heureuses initiatives, les modules d'administration, du tuteur et de l'élève n'ont pas non plus transcendé leur cadre de fonctionnement prévu. Par contre, j'ai vraiment eu le sentiment d'assister à une explosion des capacités des dispositifs externes. Lors de mon arrivée, FORMID était connecté localement à une simulation et à un micromonde trop intrusif. Finalement le micromonde est maîtrisé, la communication est asynchrone, le dispositif tangible et la simulation sur tablette tactile fonctionnent, et le mode élève déporté également. Cette partie représente pour moi une réelle satisfaction, autant du point du résultat visible que de celui de l'architecture technique qui le supporte. Je suis également fier d'avoir pu éliminer la base de données H2, et d'avoir pu stabiliser le fonctionnement de FORMID même si j'aurais préféré ne pas avoir à supprimer les WebSockets pour cela.

Du point de vue de l'apprentissage technique, je me sens capable de postuler dans les technologies que j'ai manipulées durant ce stage. Même s'il me reste de la marge de progression dans la maîtrise de Java EE et d'AngularJS, j'en sais assez pour être productif et apprendre le reste en temps masqué. Mais je retiendrai surtout l'utilisation des composants et cadriciels. En premier lieu, j'ai été confronté à l'immense complexité de la mise en page et je serai attentif aux composants que j'utilise si j'ai à les choisir. Mais surtout, avant ce stage, je n'ai jamais pris la décision d'intégrer un composant logiciel tiers dans mes applications. J'avais peur de la dépendance induite dans mon programme, et je pensais que le temps de formation serait supérieur au temps qu'il me faudrait pour développer la fonctionnalité sans le composant. J'ai pu constater que malgré l'évolution constante des composants, et donc leur obsolescence programmée, il fallait en tirer parti pour gagner en productivité. D'autre part j'estime que si le code source est accessible le risque est acceptable.

En ce qui concerne mon rôle pendant ce stage, le travail en autonomie me correspond bien, et me permet même de prendre bien plus d'initiatives que lorsque l'encadrement est très présent. Outre les développements déjà mentionnés, j'ai bien aimé par exemple planifier une solution de branchement réseau pour l'inauguration de l'Equipex.

Dans mon idée, quitter mon emploi pour effectuer un stage signifiait que je pourrais vraiment approfondir les sujets à l'extrême, notamment les différentes technologies utilisées, ainsi que la gestion de projet. Dans la

réalité, j'y ai consacré, je pense, à peine plus de temps que si j'avais été en poste avec un chef pratiquant un style de management responsabilisant. En effet, je devais rester pragmatique et viser mes objectifs d'évolution de FORMID. Je ne pouvais donc pas me permettre de gâcher du temps en cherchant à maîtriser les 20% qui me manquent sur le cadriciel côté client, ou en maintenant des diagrammes et tableaux de bord sans utilité directe pour moi.

Je pense avoir fait preuve d'esprit d'analyse tout au long de mon stage, en étudiant les impacts de mes choix potentiels, en considérant toujours l'avenir de FORMID, et en proposant beaucoup d'idées à mes responsables. J'ai su apprendre et m'adapter à un nouvel environnement technique, Java EE côté serveur, AngularJS et de multiples composants côté client. Enfin, j'ai su inventer des solutions pour que FORMID puisse lire l'état de simulations hébergées sur des tablettes tactiles, et aussi pour que chaque auteur ou élève puisse désigner la tablette avec laquelle il travaille dans FORMID.

4.4 CONCLUSION ET PERSPECTIVES

FORMID a subi une nouvelle étape d'évolution, mais il reste évidemment beaucoup à faire. En premier lieu, il convient de poursuivre le travail sur les praxéologies pour guider l'auteur et informer le tuteur. Ensuite, le déploiement dans certaines écoles à but d'expérimentation mènera logiquement à gérer plusieurs dispositifs externes simultanément, et induira donc l'évolution du système en place. Le module auteur peut évoluer pour faciliter encore l'écriture des évaluations en Java, et pour permettre à l'auteur d'agir toujours plus en détail sur des variables.

Le produit doit être laissé dans un état stable et fonctionnel, puisqu'il n'y a pas d'intervenant en continu. La transmission d'informations se fait donc par le biais de documentations claires qui décrivent les différentes composantes de cet état final.

La suite logicielle FORMID me semble en conclusion être en bonne santé. Elle repose sur des bases solides. De plus, elle est capable de se connecter à de multiples dispositifs externes, ce qui permet de donner une valeur toujours plus grande à la conception du scénario pédagogique qui pilote l'ensemble. De plus, avec la visibilité de FORMID qui augmente, cela peut donner naissance à de nouveaux projets très prometteurs.

CONCLUSION

Le logiciel FORMID est une base de travail très intéressante tant techniquement que fonctionnellement. Les manques fonctionnels ont pour la plupart été comblés, et les anomalies majeures corrigées. Tous les objectifs n'ont pas pu être atteints, notamment en raison de la panne du serveur de praxéologies. Cependant, une évolution majeure de la gestion des dispositifs externes a pu voir le jour en lieu et place. Cet aspect de connectivité accrue permet la mise en pratique de scénarios pédagogiques dans de multiples situations, qui assure des idées d'évolution non biaisées par un trop petit nombre de cas d'application.

Ma présence au sein de l'équipe MeTAH du Laboratoire d'Informatique de Grenoble a été vraiment fructueuse et instructive. J'ai su mettre mon esprit d'initiative, mes capacités d'autonomie et ma communication au service de mes responsables pour mener à bien mes missions. La confiance qui m'a été accordée m'a permis de m'épanouir dans les réalisations, et mes idées ont pu trouver le terreau nécessaire à leur épanouissement. Il s'agit d'un cercle vertueux, dans lequel le sentiment de confiance permet de prendre des initiatives qui font évoluer le produit tout en respectant sa philosophie.

BIBLIOGRAPHIE

[AlsaCreations15] AlsaCreations [en ligne]. Disponible sur : http://www.alsacreations.com/tuto/lire/610-Mise-en-page-CSS-avancee-grace-a-la-propriete-display.html (consulté en Octobre 2015).

[Angular UI Grid15] Angular UI Grid [en ligne]. Disponible sur : http://ui-grid.info/ (consulté en Septembre 2015).

[AngularJS15] GOOGLE. AngularJS [en ligne]. Disponible sur : https://angularjs.org/ (consulté en Septembre 2015).

[Apache Maven15] Apache Maven [en ligne]. Disponible sur : https://maven.apache.org/ (consulté en Septembre 2015).

[Atmosphere15] Atmosphere [en ligne]. Disponible sur : https://github.com/Atmosphere/atmosphere (consulté en Septembre 2015).

[BACHELET] BACHELET Rémi. Gestion des risques [en ligne]. Disponible sur : http://rb.ec-lille.fr/l/Gestion_risques/Gestion_des_risques_Demarche.pdf (consulté en Octobre 2015).

[Bootstrap15] Bootstrap [en ligne]. Disponible sur : http://getbootstrap.com/ (consulté en Septembre 2015).

[CAGNAT06] CAGNAT J-M. , "Projet FORMID - Les composants logiciels : descriptions et mode d'emploi," LIG, 2006.

[CHAACHOUA14] CHAACHOUA Hamid. La praxéologie comme modèle didactique pour la problèmétique EIAH. Etude de cas : la modélisation des connaissances des élèves. Technology for Human Learning [en ligne]. Janvier 2014, p. 121. Disponible sur : https://tel.archives-ouvertes.fr/tel-00922383/document (consulté en Septembre 2015).

[CHEVALLARD] CHEVALLARD Yves. Analyse des pratiques enseignantes et didactique des mathématiques : l'approche anthopologique [en ligne]. Disponible sur : http://yves.chevallard.free.fr/spip/spip/IMG/pdf/Analyse_des_pratiques_enseignantes.pdf (consulté en Septembre 2015).

[CodeMirror15] CodeMirror [en ligne]. Disponible sur : https://codemirror.net/ (consulté en Septembre 2015).

[Coding Horror15] Coding Horror [en ligne]. Disponible sur : http://blog.codinghorror.com/coding-without-comments/ (consulté en Septembre 2015).

[ECHTERBILLE14] ECHTERBILLE Patrick , "Evolution de la suite logicielle FORMID pour la conception, l'exécution et le suivi de situations d'apprentissage à distance," CNAM, Grenoble, Mémoire 2014.

[ECHTERBILLE14] ECHTERBILLE Patrick , "Document d'Architecture Technique Formid," 2014.

[Ember.js15] Ember.js [en ligne]. Disponible sur : http://emberjs.com/ (consulté en Octobre 2015).

[Font Awesome15] Font Awesome [en ligne]. Disponible sur : http://fortawesome.github.io/Font-Awesome/ (consulté en Septembre 2015).

[Framablog15] Framablog [en ligne]. Disponible sur : http://framablog.org/2009/11/21/commentaires-code-source-programmation/ (consulté en Septembre 2015).

[FusionForge15] FusionForge [en ligne]. Disponible sur : https://fusionforge.org/ (consulté en Septembre 2015).

[GAMMA94] GAMMA Helm, Johnson, Vlissides. Design Patterns - Elements of Reusable Object-Oriented Software. 1st. , 1994, 395 p.
ISBN 978-0201633610

[git15] git [en ligne]. Disponible sur : https://git-scm.com/ (consulté en Septembre 2015).

[GUÉRAUD11] GUÉRAUD Lejeune. Une approche auteur pour la scénarisation et le suivi de situations d'apprentissage. Environnements Informatiques pour l'Apprentissage Humain, Mons [en ligne]. 2011, p. 6.

[H2Database15] H2Database [en ligne]. Disponible sur : http://www.h2database.com (consulté en Septembre 2015).

[Hibernate ORM15] Hibernate ORM [en ligne]. Disponible sur : http://hibernate.org/orm/ (consulté en Septembre 2015).

[HOUSSAYE00] HOUSSAYE Jean. Le triangle pédagogique [en ligne]. , 2000 Disponible sur : http://www.anim.ch/pxo3_02/pxo_content/medias/jean_houssaye_triange_pedagogique.pdf (consulté en Novembre 2015).

[HTML5 sessionStorage15] HTML5 sessionStorage [en ligne]. Disponible sur : http://code.google.com/p/sessionstorage/ (consulté en Septembre 2015).

[html5shiv15] html5shiv [en ligne]. Disponible sur : https://github.com/aFarkas/html5shiv (consulté en Septembre 2015).

[JavaScript Load Image15] JavaScript Load Image [en ligne]. Disponible sur : https://blueimp.github.io/JavaScript-Load-Image/ (consulté en Septembre 2015).

[JavaScript-Canvas-to-Blob15] JavaScript-Canvas-to-Blob [en ligne]. Disponible sur : https://github.com/blueimp/JavaScript-Canvas-to-Blob (consulté en Septembre 2015).

[Jersey15] Jersey [en ligne]. Disponible sur : https://jersey.java.net/ (consulté en Septembre 2015).

[JPEXS Free Flash Decompiler15] JPEXS Free Flash Decompiler [en ligne]. Disponible sur : https://www.free-decompiler.com/flash/ (consulté en Octobre 2015).

[jQuery atmosphere15] jQuery atmosphere [en ligne]. Disponible sur : https://github.com/Atmosphere/atmosphere/wiki/jQuery.atmosphere.js-atmosphere.js-API (consulté en Septembre 2015).

[jQuery custom content scroller15] jQuery custom content scroller [en ligne]. Disponible sur : http://manos.malihu.gr/jquery-custom-content-scroller/ (consulté en septembre 2015).

[jQuery File Upload15] jQuery File Upload [en ligne]. Disponible sur : https://blueimp.github.io/jQuery-File-Upload/ (consulté en Septembre 2015).

[jQuery UI Layout Plug-in15] jQuery UI Layout Plug-in [en ligne]. Disponible sur : http://layout.jquery-dev.com/index.cfm (consulté en Septembre 2015).

[jQuery UI15] jQuery UI [en ligne]. Disponible sur : https://jqueryui.com/ (consulté en Septembre 2015).

[jQuery15] jQuery [en ligne]. Disponible sur : http://jquery.com/ (consulté en Septembre 2015).

[jsTree15] jsTree [en ligne]. Disponible sur : http://www.jstree.com/ (consulté en Septembre 2015).

[LIG15] LIG [en ligne]. Disponible sur : https://www.liglab.fr/presentation/organigramme (consulté en Septembre 2015).

[MDN server-sent events15] MDN server-sent events [en ligne]. Disponible sur : https://developer.mozilla.org/fr/docs/Server-sent_events/Using_server-sent_events (consulté en Septembre 2015).

[MDN XMLHttpRequest] MDN XMLHttpRequest [en ligne]. Disponible sur : https://developer.mozilla.org/fr/docs/Web/API/XMLHttpRequest (consulté en).

[MeSpeak15] MeSpeak [en ligne]. Disponible sur : http://www.masswerk.at/mespeak/ (consulté en Octobre 2015).

[MeTAH15] MeTAH [en ligne]. Disponible sur : http://metah.imag.fr/ (consulté en Septembre 2015).

[Miximum15] Miximum [en ligne]. Disponible sur : http://www.miximum.fr/un-commentaire-a-quoi-ca-sert.html (consulté en Septembre 2015).

[Moment.js15] Moment.js [en ligne]. Disponible sur : http://momentjs.com/ (consulté en Septembre 2015).

[MORLEY08] MORLEY Chantal. Management d'un Projet Système d'Information. 6th. , 2008, 458 p. ISBN 978-2-10-052088-6

[MSDN Async Await15] MSDN Async Await [en ligne]. Disponible sur : https://msdn.microsoft.com/fr-fr/library/hh191443.aspx (consulté en Octobre 2015).

[MSDN15] MICROSOFT. MSDN [en ligne]. Disponible sur : https://msdn.microsoft.com/en-us/library/ie/ms535263%28v=vs.85%29.aspx (consulté en Mars 2015).

[Multicast DNS15] Multicast DNS [en ligne]. Disponible sur : http://www.multicastdns.org/ (consulté en Octobre 2015).

[PostgreSQL15] PostgreSQL [en ligne]. Disponible sur : http://www.postgresql.org/ (consulté en Septembre 2015).

[REENSKAUG07] REENSKAUG Trygve , "The original MVC reports," Dept. of Informatics, University of Oslo, Oslo, 2007.

[ResponsiveVoice.JS15] ResponsiveVoice.JS [en ligne]. Disponible sur : http://responsivevoice.org/ (consulté en Septembre 2015).

[restangular15] restangular [en ligne]. Disponible sur : https://github.com/mgonto/restangular (consulté en Septembre 2015).

[Spring15] Spring [en ligne]. Disponible sur : https://spring.io/ (consulté en Septembre 2015).

[StackOverflow15] StackOverflow [en ligne]. Disponible sur : http://stackoverflow.com/ (consulté en Octobre 2015).

[StackOverflow15] StackOverflow [en ligne]. Disponible sur : http://stackoverflow.com/questions/83073/why-not-use-tables-for-layout-in-html (consulté en Octobre 2015).

[SysInternals15] SysInternals [en ligne]. Disponible sur : https://technet.microsoft.com/en-us/sysinternals/bb842062.aspx (consulté en Octobre 2015).

[TodoMVC15] TodoMVC [en ligne]. Disponible sur : http://todomvc.com/ (consulté en Septembre 2015).

[TPElec15] TPElec [en ligne]. Disponible sur : http://tpelec.imag.fr/ (consulté en Septembre 2015).

[UAE4ALL15] UAE4ALL [en ligne]. Disponible sur : http://chui.dcemu.co.uk/uae4all.html (consulté en Septembre 2015).

[UI Bootstrap15] UI Bootstrap [en ligne]. Disponible sur : https://angular-ui.github.io/bootstrap/ (consulté en Septembre 2015).

[Underscore.js15] Underscore.js [en ligne]. Disponible sur : http://underscorejs.org (consulté en Septembre 2015).

[UnderTracks15] METAH. UnderTracks [en ligne]. Disponible sur : https://undertracks.imag.fr/php/ (consulté en Octobre 2015).

[W3C Server-Sent Events15] W3C Server-Sent Events [en ligne]. Disponible sur : http://www.w3.org/TR/2011/WD-eventsource-20111020/#server-sent-events-intro (consulté en Octobre 2015).

[w3schools.com15] w3schools.com [en ligne]. Disponible sur : http://www.w3schools.com/html/html_responsive.asp (consulté en Octobre 2015).

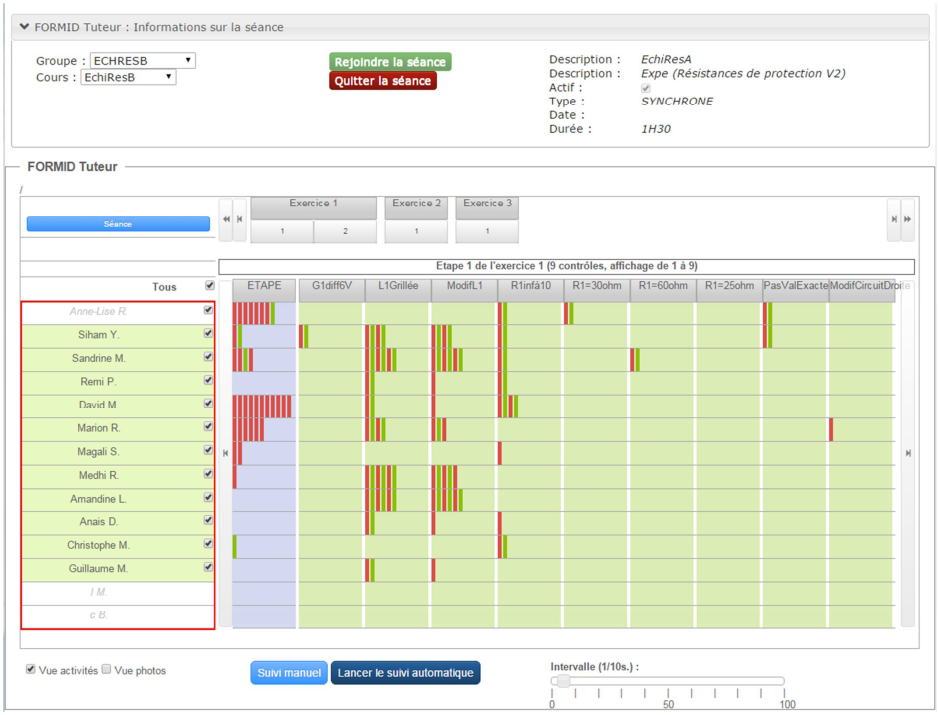
[WebSocket15] WebSocket [en ligne]. Disponible sur : https://www.websocket.org/ (consulté en Septembre 2015).

[Weld15] Weld [en ligne]. Disponible sur : http://weld.cdi-spec.org/ (consulté en Septembre 2015).

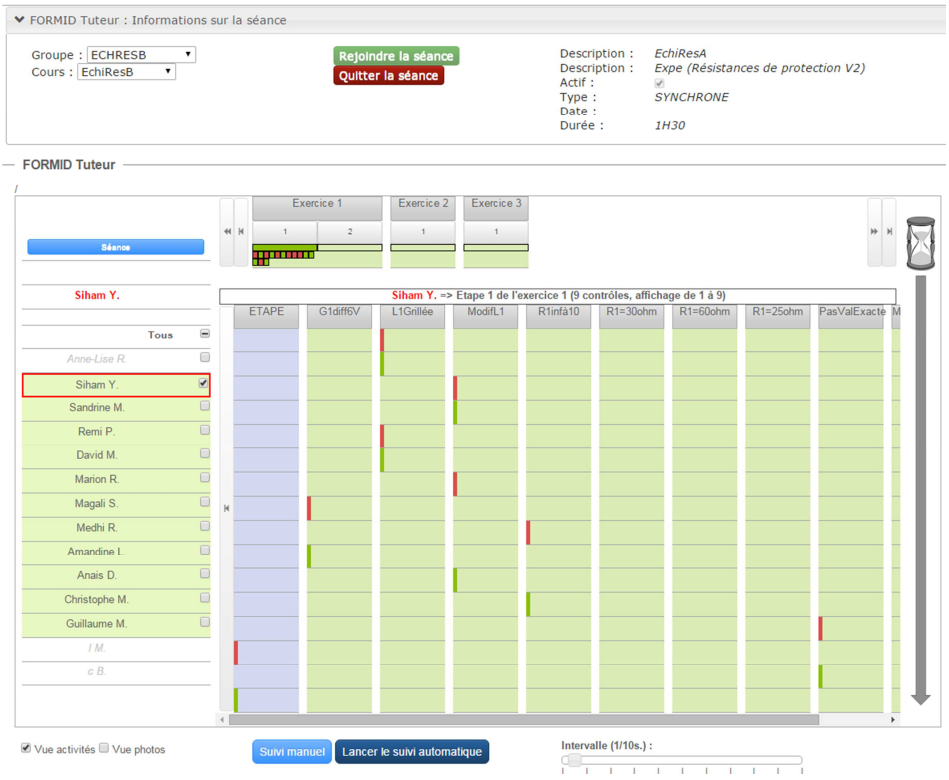
ANNEXES

1 VUES DU MODE TUTEUR

La vue globale est présentée Figure 11 page 12. La deuxième vue entre dans le détail de tous les contrôles déclenchés pour une étape d'un exercice donné :



La troisième vue affiche la chronologie des déclenchements de contrôles pour un élève sur une étape donnée :



2 INAUGURATION DE L'EQUIPEX AMIQUAL4HOME

INVITATION

Inauguration Equipex Amiqual4Home

Vendredi 5 juin 2015 /14h00 - 17h00

51 avenue Jean Kuntzmann - 38330 Montbonnot Saint Martin

14h00 - Inauguration de l'Equipex Amiqual4Home

avec l'intervention de

- **James Crowley,** Professeur Grenoble INP, Responsable de l'EquipEx Amiqual4home
- **Jean-Pierre Verjus,** Président du conseil d'administration de Digital Grenoble
- **François James,** Agence Nationale de la Recherche
- **Brigitte Plateau,** Administrateur Général de Grenoble INP
- **Sophie Jullian,** Direction Régionale à la Recherche et à la Technologie

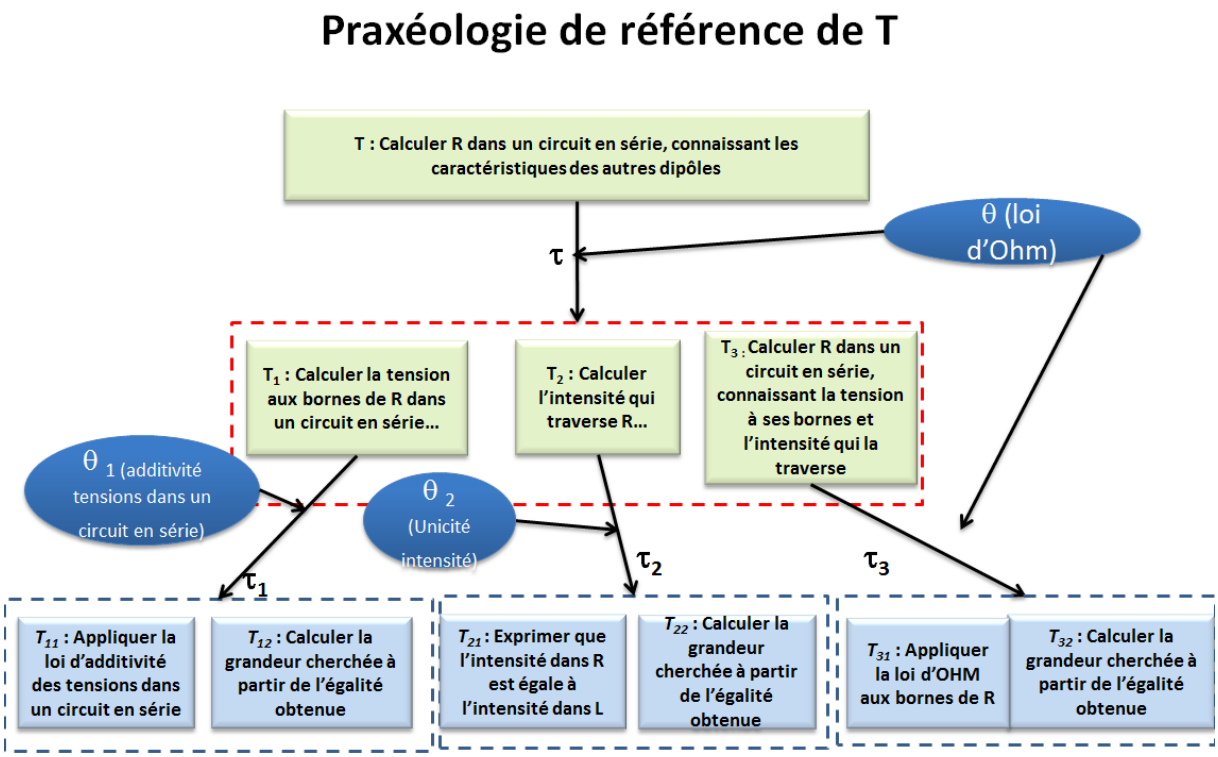
La cérémonie sera suivie de visites et de démonstrations

CONTACTS :
Amiqual4Home - Catherine Bessière - Tél. : 04 76 61 53 61 - catherine.bessiere@inria.fr

Inauguration du 05 juin 2015 du bâtiment Amiqual4Home pour l'Equipex.
Stand tenu par : Nadine Mandran (Ingénieur SHS – CRNS, LIG), Anne Lejeune (Maître de conférences), Jade Thiriat (stagiaire ENSIMAG), Emmanuel Létondor (stagiaire Cnam).

3 PRAXÉOLOGIE DE RÉFÉRENCE POUR UN CIRCUIT ÉLECTRIQUE

Ce diagramme représente le cas d'étude conçu pour la première mise en œuvre des praxéologies dans FORMID. Il ne s'agit que d'un extrait basé sur un seul type de tâche et servant de base de travail. La théorie d'un modèle praxéologique est expliquée dans la partie 2.1.5.



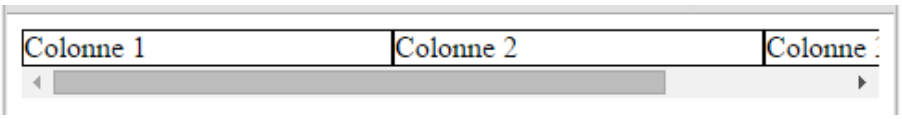
Les types de tâches T (rectangles pleins) sont réalisés par des techniques τ (Tau) (flèches notées τ et rectangles pointillés). Chaque technique, justifiée par une technologie θ (thêta), peut être réalisée par des types de tâches. Les types de tâche du bas sont élémentaires, donc ils ne peuvent plus être réalisés par une technique.

4 MAQUETTE DE COMPORTEMENT DÉSIRÉ POUR RÉSOUDRE UN PROBLÈME DE MISE EN PAGE

Pour résoudre un problème de mise en page, il m'arrive de créer une petite maquette pour atteindre le résultat attendu. Ici l'objectif était de résoudre le problème du module tuteur dans lequel les colonnes d'exercice passent en dessous des autres lorsque la largeur du navigateur est insuffisante pour les afficher.

```
<html>
    <head>
        <style type="text/css">
div.table {
    display:table;
    width:100%;
}

div.column
{
    border:1px solid #000000;
    display:table-cell;
    width:200px;
}
        </style>
    </head>
    <body>
        <div style="overflow:auto;">
            <div class="table" style="width:600px;">
                <div class="column">
                    Colonne 1
                </div>
                <div class="column">
                    Colonne 2
                </div>
                <div class="column">
                    Colonne 3
                </div>
            </div>
        </div>
    </body>
</html>
```



Ce résultat convient parfaitement. D'une part la taille des colonnes respecte la consigne de 200 pixels, puis les colonnes qui n'entrent pas dans l'espace disponible ne passent pas à la ligne (les colonnes sont en fait des éléments "`<div>`"), et la barre de défilement apparaît.

5 MODES DE COMMUNICATION DE FORMID AVEC LES DISPOSITIFS EXTERNES

Historiquement, c'est *FORMID* qui a toujours interrogé les simulations et micromondes liés, dont les principaux sont *TPElec* pour l'électricité et *TPNum* pour la numération à l'aide de billes (Cf. premier diagramme). Le deuxième diagramme de séquence UML montre la proposition d'évolution de *FORMID* pour qu'il puisse se mettre également à l'écoute des dispositifs externes.

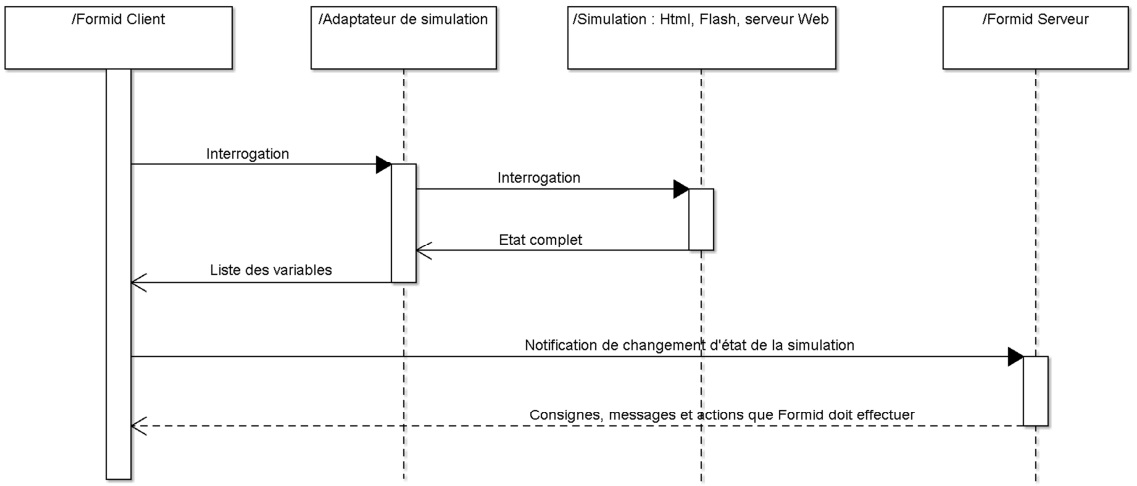


Diagramme de séquence UML montrant *FORMID* en mode **actif** qui contacte le dispositif externe de manière synchrone.

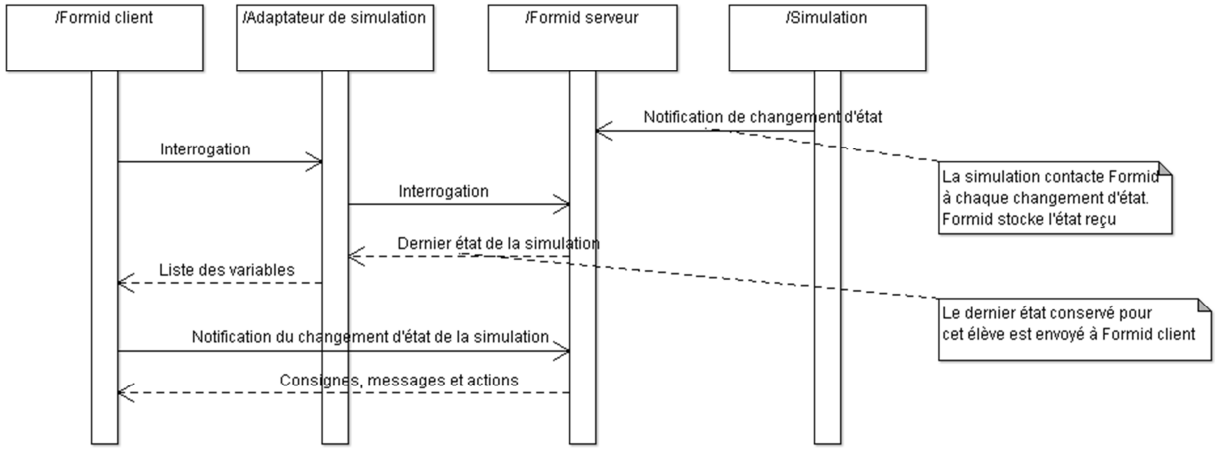
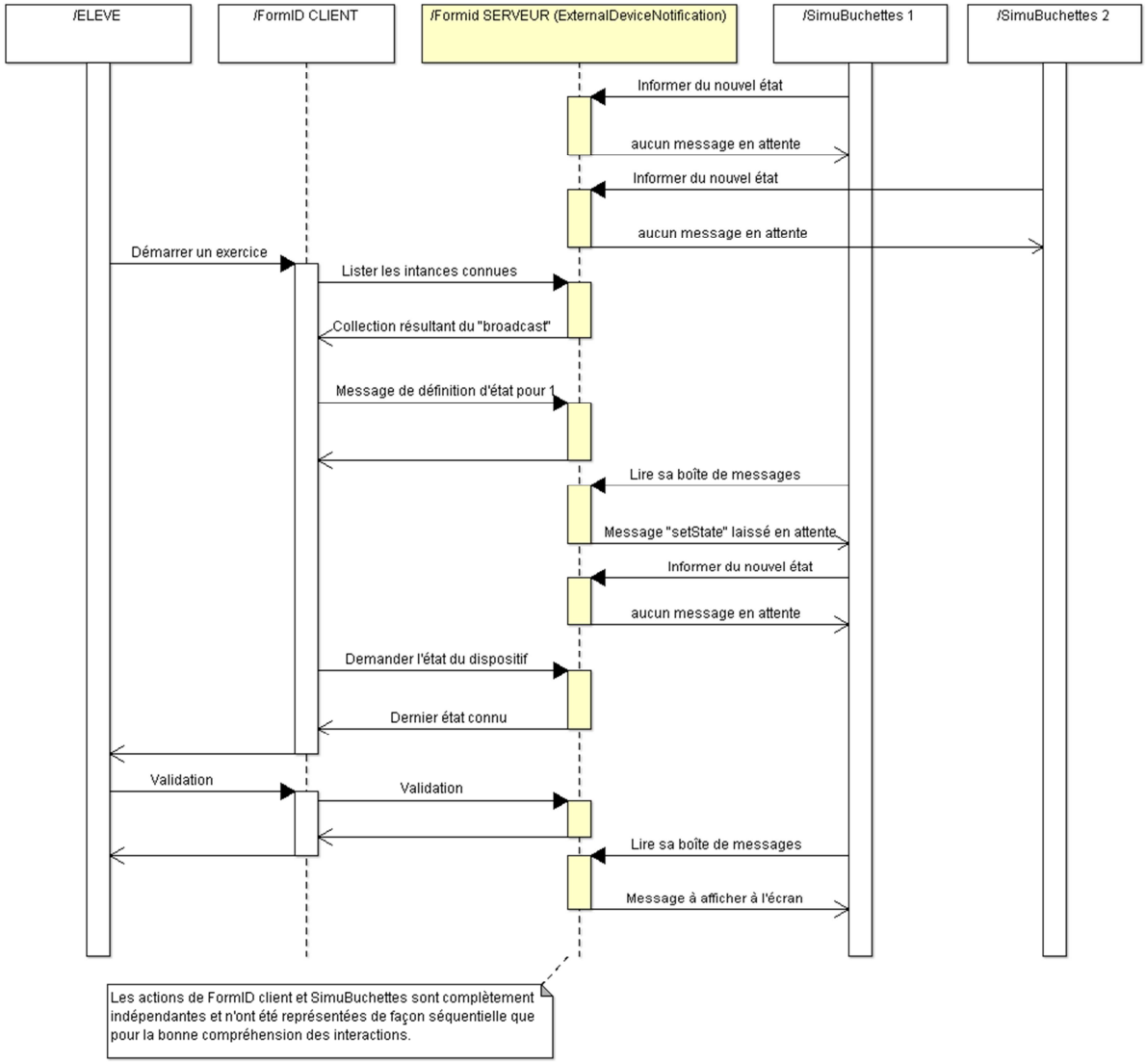


Diagramme de séquence UML montrant *FORMID* en mode **passif**, contacté par le dispositif externe, avec communication asynchrone.

6 FORMID EN MODE PASSIF PEUT ÉCOUTER LES DISPOSITIFS EXTERNES

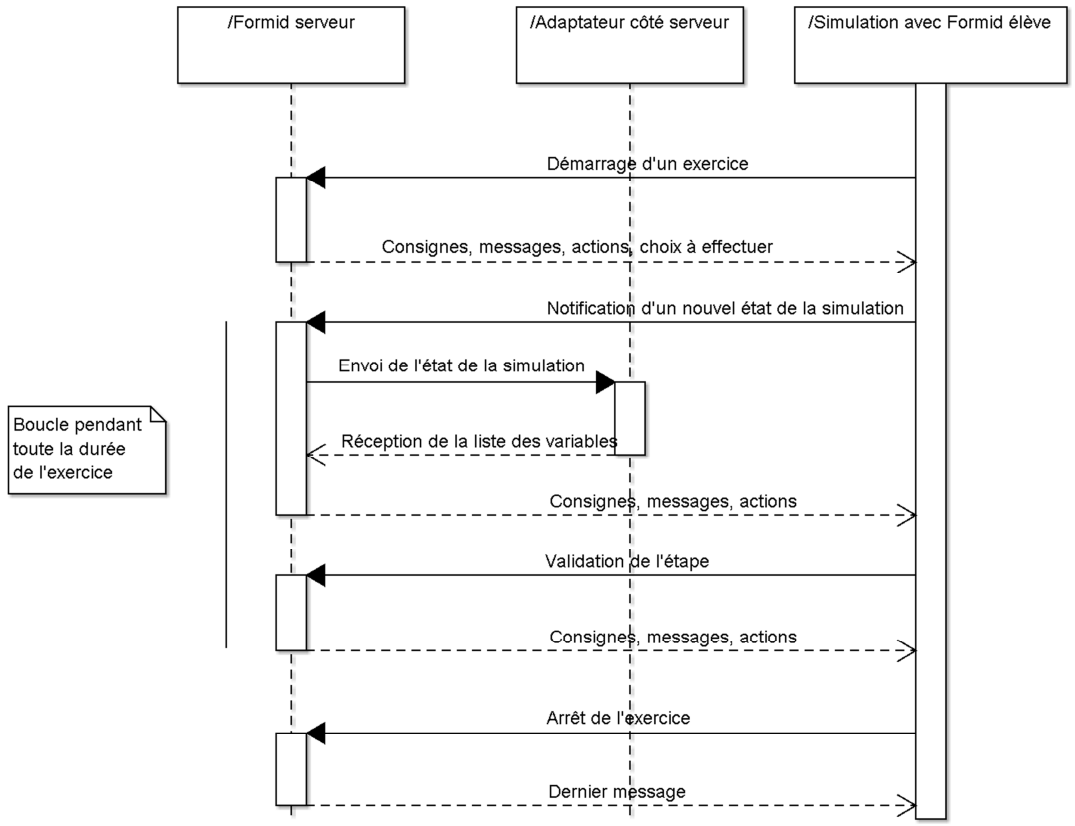
Le mode passif de FORMID permet de ne pas interroger un dispositif externe, mais d'attendre qu'il nous notifie. Ce mécanisme a été pensé pour la liaison à des tablettes tactiles qui hébergent une simulation.



Ce diagramme est la version évoluée et détaillée du deuxième diagramme de l'Annexe 5, avec une nouvelle considération : la gestion de multiples exemplaires du dispositif externe.

7 MODE ÉLÈVE DÉPORTÉ

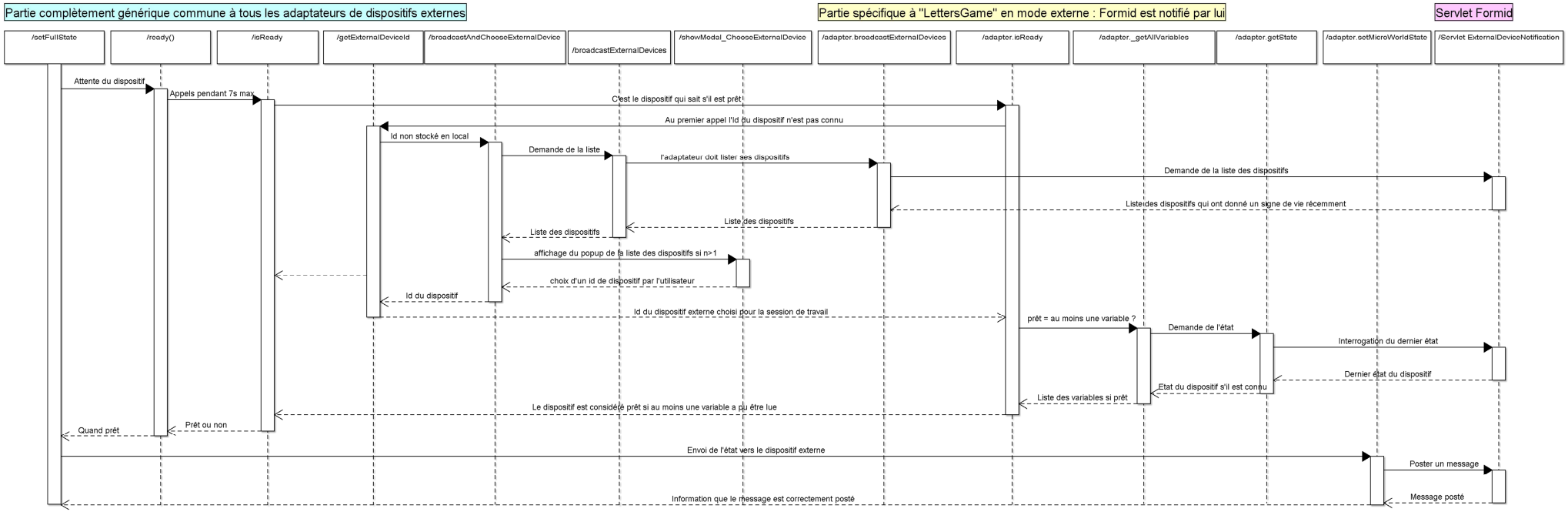
Le mode élève déporté consiste à permettre de s'affranchir de l'utilisation de la page web de *FORMID* pour exécuter un scénario pédagogique. Le dispositif externe doit donc être capable d'afficher les consignes de l'exercice et doit présenter des boutons d'action, dont a minima un bouton de validation d'étape.



Formid élève déporté au sein d'un dispositif externe "actif", qui notifie FORMID, lequel, lui, est en mode passif.

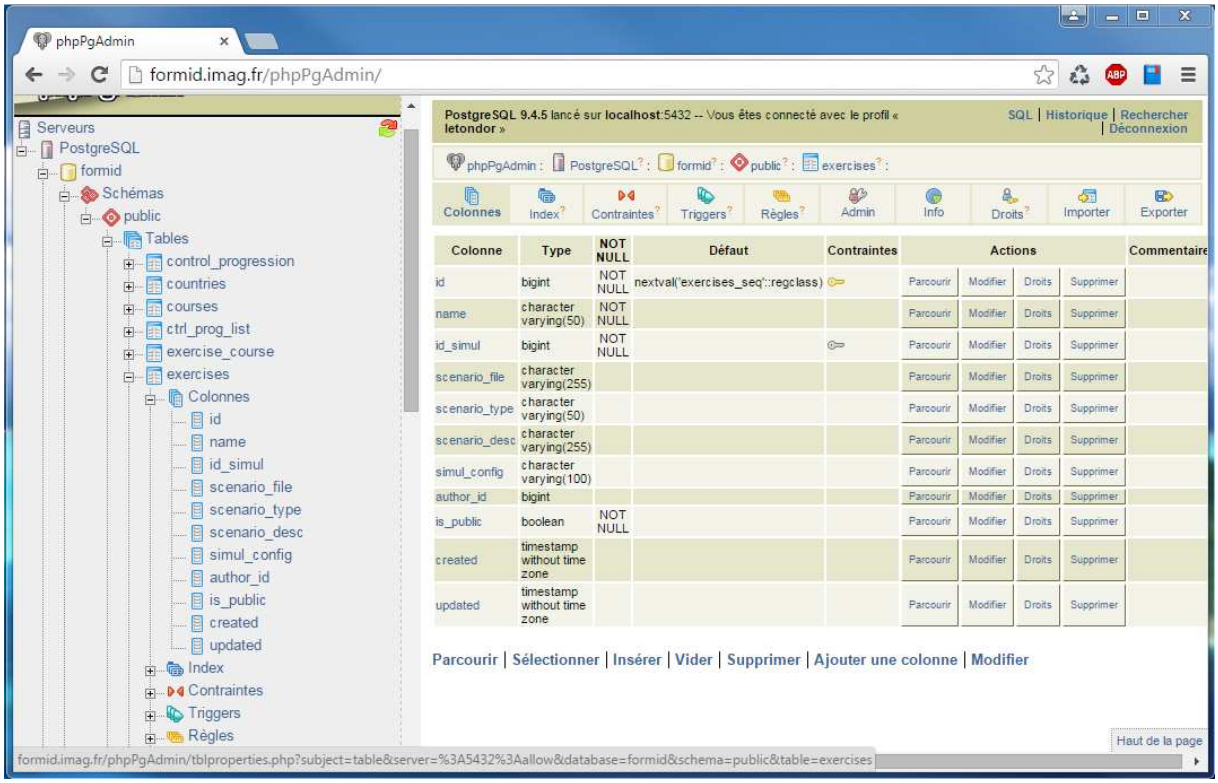
8 COMMUNICATION AVEC DES DISPOSITIFS EXTERNES MULTIPLES

Le diagramme de séquence UML ci-dessous se situe au moment du démarrage d'un exercice par un élève. FORMID attend que le dispositif externe soit prêt pour envoyer l'état initial et ainsi être apte à exécuter l'exercice. Comme tous les appels sont asynchrones, ils sont tous réalisés dans le programme par une « promesse ». Ce diagramme illustre donc également le grand empilement des appels en attente. Nous pouvons constater qu'il existe jusqu'à neuf promesses imbriquées.



9 BASE DE DONNÉES POSTGRESQL

La base de données a été migrée de « H2 Database » vers « PostgreSQL ». Voici l'interface d'administration web « phpPgAdmin » :



10 DOCUMENTS RÉDIGÉS

Voici la liste complète des documents que j'ai pris l'initiative de rédiger pendant mes 9 mois de travail sur la suite FORMID :

Nom du document	Date de création	Nombre de pages	Contenu
ConfigPourFormidEtJade_2015-06-05.docx	05/06/2015	1	Diagramme du réseau prévu pour l'inauguration du bâtiment Amiqual4Home, version 1
ConfigPourFormidEtJade2_2015-06-05.docx	05/06/2015	1	Deuxième et dernière version du réseau pour l'inauguration du bâtiment Amiqual4Home.
Dispositifs Externes.xlsx	30/07/2015	1	Tableau Excel récapitulatif des dispositifs externes, pour identifier leurs points communs et différences de traitement.
FORMID Ajout d'un dispositif externe.docx	08/07/2015	8	*FORMID* a vocation à accueillir toujours plus de dispositifs externes. Ce document décrit comment en intégrer un nouveau. Les explications passent par l'inscription dans la base de données, les fichiers de configuration, et décrit comment l'adaptateur de communication, matérialisé par une page JSP, doit être développé.
FORMID Auteur - publication.docx	23/04/2015	2	Ce document technique décrit le cheminement des fichiers d'un scénario tel que je l'ai implémenté lorsqu'il est compilé puis publié.
FORMID Dialogue avec les dispositifs externes.docx	06/07/2015	16	Cette documentation fait également office de spécification. Elle décrit la manière qu'avait FORMID de dialoguer avec un dispositif externe à la date de création du document, ainsi que mes propositions d'évolution : un mode inversé dans lequel le dispositif externe notifierait FORMID de son changement d'état, et la proposition d'un mode élève déporté.
FORMID Dispositifs externes multiples.docx	07/08/2015	7	A partir du moment où le dispositif externe n'est pas instancié par FORMID, la possibilité existe qu'il en existe plusieurs instances. Ce document décrit donc l'évolution que j'ai faite dans FORMID pour gérer le choix d'un dispositif externe parmi plusieurs, des points de vue théorique et technique. Il contient le diagramme de séquence correspondant.
FORMID Eleve.docx	24/04/2015	1	Report d'analyse technique du lancement d'un exercice en mode élève.
FORMID Etat des lieux début mars 2015.docx	04/03/2015	5	Un stage d'IUT a été réalisé par Timothée Lemaire début 2015. Son code n'avait pas été poussé dans le dépôt GIT mis en place par Patrick Echterbille, mon prédécesseur Cnam. Ce document visait à établir s'il y avait un risque à intégrer le travail réalisé sur TPNum. Verdict : risque minime.
FORMID Explications de LettersGame.docx	01/09/2015	2	Pour que FORMID puisse évoluer dans le sens spécifié par "FORMID Dialogue avec les dispositifs externes.docx", il me fallait un dispositif externe. Or le lien avec l'équipe de la simulation sur tablette tactile "*SimuBûchettes*" n'avait pas encore été entériné. J'ai donc développé une petite simulation qui permet de déplacer des lettres et des chiffres sur des plateaux comme support. Cette simulation, "jeux de lettres", fonctionne donc suivant 3 modes qu'il a fallu expliquer : interne à FORMID, externe, externe avec mode élève déporté.

Nom du document	Date de création	Nombre de pages	Contenu
FORMID Implementer Dispositif Externe multiple avec mode Elève.docx	08/09/2015	12	Les évolutions techniques de FORMID permettant de gérer des dispositifs externes multiples qui notifient FORMID ont impliqué la création d'un service Web de communication. Les paramètres et les trames échangées devaient être décrits en détail pour permettre l'implémentation réelle future dans *SimuBûchettes* ou d'autres dispositifs externes.
FORMID Mise en route poste de développement.docx	03/03/2015	15	Lorsque j'ai commencé mon stage, j'ai pris la décision de ne pas travailler dans le dossier des sources de mon prédécesseur, pour toujours avoir une version originale que je puisse compiler. En plus, le fait d'avoir créé mon propre compte Windows m'a obligé à re-paramétrer l'IDE. Ce document contient donc l'ensemble des opérations d'installation et de paramétrage : Flash, NetBeans, Java, JBoss, H2Database (rendu obsolète par la suite), GIT, Maven 2, et choix des composants additionnels de l'IDE.
FORMID Mode élève déporté - contraintes.docx	04/09/2015	5	Penser à un mode élève qui puisse être piloté directement par le dispositif externe, comme une tablette tactile, demande à trouver une solution d'intrusion minimale. Donc conserver le fonctionnel au sein de FORMID tout en offrant la possibilité de faire choisir un exercice ou autre chose dans la tablette.
FORMID Nouvel arrivant.docx	11/09/2015	16	J'ai conçu ce document en pensant à un bureau : ce que l'on a sur le bureau, ce sont toutes les informations utiles, les URLs, les mots de passe, l'arborescence de travail, les outils. Il indique surtout ce qu'il faut consulter pour approfondir chaque partie : 13 documents cités en référence, dont 2 qui ne sont pas de moi mais de Patrick Echterbille.
FORMID solution d'identification d'un dispositif externe.docx	03/08/2015	6	Dans le cas des dispositifs externes multiples qui notifient FORMID de leur changement d'état, se posent de nombreuses questions, dont celle du couplage d'un dispositif avec une session de travail élève ou auteur. C'est un document ne contenant que des questions et des hypothèses, pour identifier l'étendue des problématiques à résoudre.
FORMID Tangible - besoins v1.0.docx	10/03/2015	12	Pour être capable de valider la spécification de développement du service REST pour la partie tangible, j'ai estimé nécessaire de passer en revue tous nos besoins pour les mettre en adéquation avec la proposition.
FORMID_RetryStep.xlsx	17/07/2015	1	Ce tableau Excel répertorie l'ensemble du traitement des états d'un dispositif externe pour les fonctions "réessayer l'étape" et "sauter l'étape". Par exemple, si l'élève souhaite réessayer l'étape, devons-nous lui mettre son dispositif dans l'état initial de l'étape tel que défini par l'auteur, ou dans l'état dans lequel il était ? Il sert également de base de test et explique les choix qui ont été faits.
NoteCard_AngularJS.docx	09/03/2015	16	Fiche explicative de *AngularJS*, se voulant un aide-mémoire sur les grands concepts, leur mise en œuvre, et les mécanismes communs. Elle permettra à un nouveau venu dans le domaine d'appréhender bien plus vite les concepts.
NoteCard_Bootstrap.docx	16/03/2015	3	Notions principales pour savoir si le code Html que l'on voit est du BootStrap ou non, surtout concernant la mise en page.

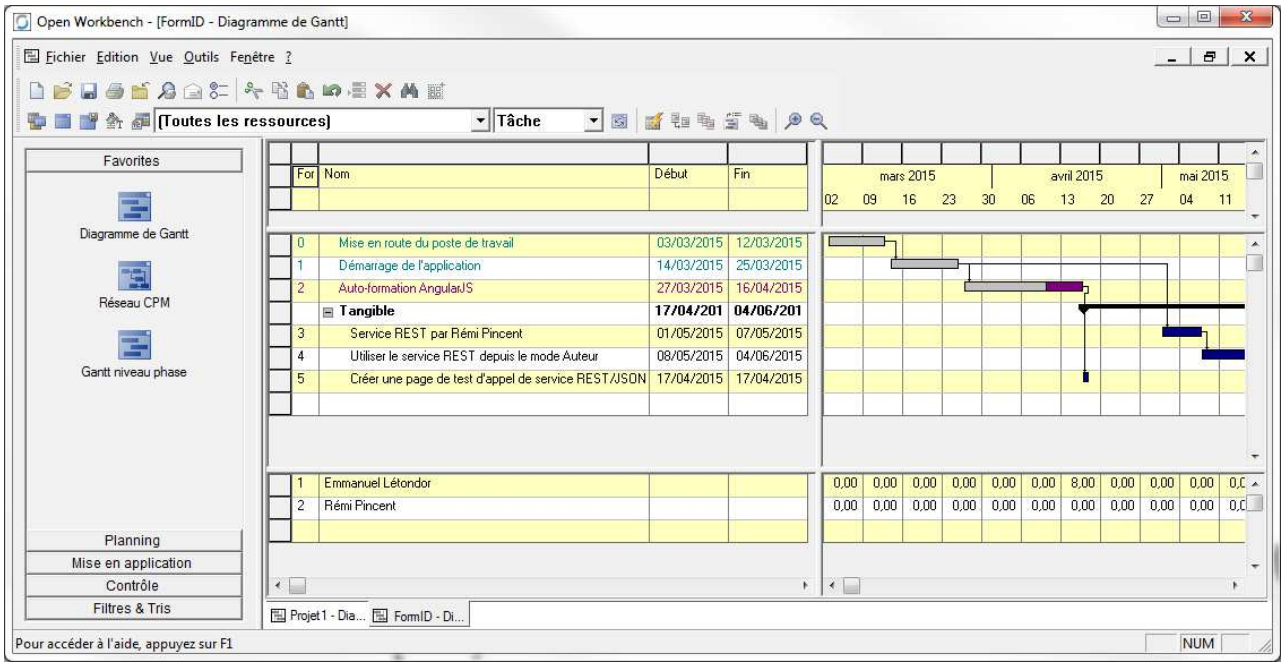
Nom du document	Date de création	Nombre de pages	Contenu
NoteCard_Code comments.docx	04/08/2015	7	Je ne suis pas du tout adepte du concept selon lequel un code suffisamment clair se suffit à lui-même. Pas plus que celui selon lequel indiquer ses modifications ne sert à rien depuis l'avènement du contrôle de code source. Je commente tout, partout, et j'explique ma façon de travailler. En synthèse, les commentaires me paraissent indispensables pour expliquer le "pourquoi".
NoteCard_Code.docx	17/03/2015	8	Fiche technique contenant quelques astuces pour notamment localiser des anomalies dans son code.
NoteCard_DatabaseH2.docx	03/05/2015	15	La base de données "H2" mise en place par mon prédécesseur m'a donné beaucoup de mal, souvent. En effet, ce n'est pas un service mais un fichier java JAR à lancer, dont il n'est pas possible de contrôler les instances, et qui n'a pas de journal détaillé. L'accès à distance n'est de plus pas possible, ce qui oblige à quitter les instances lancées pour mettre à jour le serveur. Ce document est rendu obsolète par la migration à la base "PostgreSQL".
NoteCard_Déploiement.docx	17/04/2015	3	Très important, explique comment déployer une application FORMID qui fonctionne en local sur le poste Windows vers le serveur Linux CentOS, et comment contrôler les services Apache et Wildfly.
NoteCard_FAQ.docx	22/04/2015	1	Solutions à des problèmes rencontrés, surtout dans NetBeans : plus de mémoire, fermeture inopinée, etc.
NoteCard_GIT utilisation.docx	09/03/2015	4	Aide-mémoire concernant l'utilisation de GIT. Ne contient rien de spécifique à FORMID.
NoteCard_Html5.docx	13/03/2015	3	Prémices d'un aide-mémoire sur Html5, peu utile.
NoteCard_i18n.docx	14/04/2015	9	Le code contient une notion de "i18n" qui laisse penser à l'utilisation d'un composant additionnel. Or ce terme signifie simplement "internationalisation", et désigne le code Java et JavaScript mis en place pour gérer le multilinguisme. Après analyse, j'ai expliqué où et comment cela fonctionne.
NoteCard_Infrastructure LIG.docx	08/06/2015	3	Informations importantes données par M. Mas, lors de nos échanges par mail, concernant la logique de fonctionnement de l'infrastructure (réseau local) et des serveurs CentOS.
NoteCard_JQuery.docx	12/03/2015	6	Aide-mémoire JQuery pour ceux qui ne connaissent pas (ce qui était mon cas)
NoteCard_Layout.docx	30/03/2015	1	Contient quelques informations sur l'organisation de la mise en page de FORMID.
NoteCard_LINUX CentOS et PostGreSQL.docx	01/04/2015	29	Ce document contient l'ensemble des commandes, des chemins et des informations nécessaires pour "administrer" FORMID sur le serveur Linux CentOS. Il explique par exemple aussi comment installer et configurer PostgreSQL.
NoteCard_MaJ_Frameworks.docx	18/03/2015	7	Contient des explications de la problématique de la mise à jour des composants logiciels côté client, du travail de réorganisation que j'ai fait pour le permettre, et mes notes de mise à jour.
NoteCard_Outils.docx	14/04/2015	5	Liste tous les outils que j'ai été amené à manipuler pour travailler sur FORMID.

Nom du document	Date de création	Nombre de pages	Contenu
NoteCard_PostgreSQL.docx	13/04/2015	10	Ce document explique la migration de H2 Database vers PostgreSQL, c'est-à-dire comment faire fonctionner et administrer PostgreSQL sur le poste de développement, et faire en sorte que le déploiement fonctionne sur Linux. Devient pratiquement inutile une fois que l'on est passé en PostgreSQL.
NoteCard_TPElec.docx	14/04/2015	10	Ce recueil d'informations est issu de mon analyse fine du dialogue du dispositif externe TPElec avec FORMID, et de la décompilation du composant Flash pour en comprendre les rouages et le fonctionnement. La gestion des circuits, notamment, est difficile à cerner.
NoteCard_TPElec_Flash.docx	07/07/2015	6	Dans l'objectif de supprimer le lien fort de TPElec avec la partie serveur de FORMID pour rendre l'affectation de circuit générique et plus flexible, j'analyse le code, et explique ce que je vais faire, et ce qui serait bien de faire évoluer dans TPElec.
NoteCard_Wildfly(JBoss).docx	18/03/2015	3	Ce document contient mes explications du fonctionnement de Wildfly sur Windows. Lorsque NetBeans ne le supportait pas nativement, le déploiement se faisait d'une manière simulée sur JBoss, mais en réalité dans Wildfly. Patrick Echterbille avait expliqué la difficulté de se lier au *Debugger*. Après utilisation d'un composant, NetBeans peut directement déploier sur Wildfly et le *Debugger* fonctionne sans opération supplémentaire.
ONTOPRAX - Procédure de mise en route.docx	26/06/2015	8	Le serveur d'ontologies Ontoprax était déjà "en panne" au début de mon stage. L'ancien responsable Mohameth François étant parti, j'ai noté toutes les informations recueillies lors de nos recherches avec Cyrille Desmoulins sur comment remettre en état de fonctionnement les serveurs Apache et Virtuoso. Environnement Mac OS X.
KB-2015-001 Form File upload - Full path included or not.docx	06/03/2015	2	Fiche de la base de connaissances. Problématique spécifique Internet Explorer : lors de l'envoi d'un fichier vers le serveur, le nom indiqué pour le fichier par le formulaire contient parfois le chemin complet sélectionné, et parfois uniquement le nom du fichier.
KB-2015-002 Git Disconnected No supported authentication methods available (Public Key).docx	09/03/2015	1	Fiche de la base de connaissances. Lorsque GIT produit des erreurs d'authentification, explique comment définir sa clé privée et sur quel document s'appuyer.
KB-2015-003 Linux WebSockets problem.docx	31/08/2015	1	Fiche de la base de connaissances pour restaurer le fonctionnement des WebSockets.
FORMID Traces Tuteur.docx	25/10/2015	3	Recueil des informations comprises du modèle de données.
FORMID_Dispositifs tangibles multiples.docx	23/11/2015	5	Paramétrages à appliquer pour que FORMID puisse se connecter à plusieurs dispositifs tangibles.
FORMID Export base de données PostgreSQL.docx	24/11/2015	3	Commandes permettant d'exporter l'intégralité des données sous forme de scripts SQL.
FORMID Traces Auteur.docx	24/11/2015	9	Analyse et explications permettant à un futur intervenant d'améliorer les traces auteur.
FORMID Dispositifs tangibles.docx	25/11/2015	8	Explications, paramétrages mDNS, méthodes de diagnostic pour se connecter aux deux dispositifs tangibles actuellement en notre possession.
TOTAL	**50 documents**	**300 pages**	

11 LOGICIELS GRATUITS DE GESTION DE PROJET TESTÉS

Voici les outils trouvés lors de ma phase de recherche d'un outil de gestion de projet.

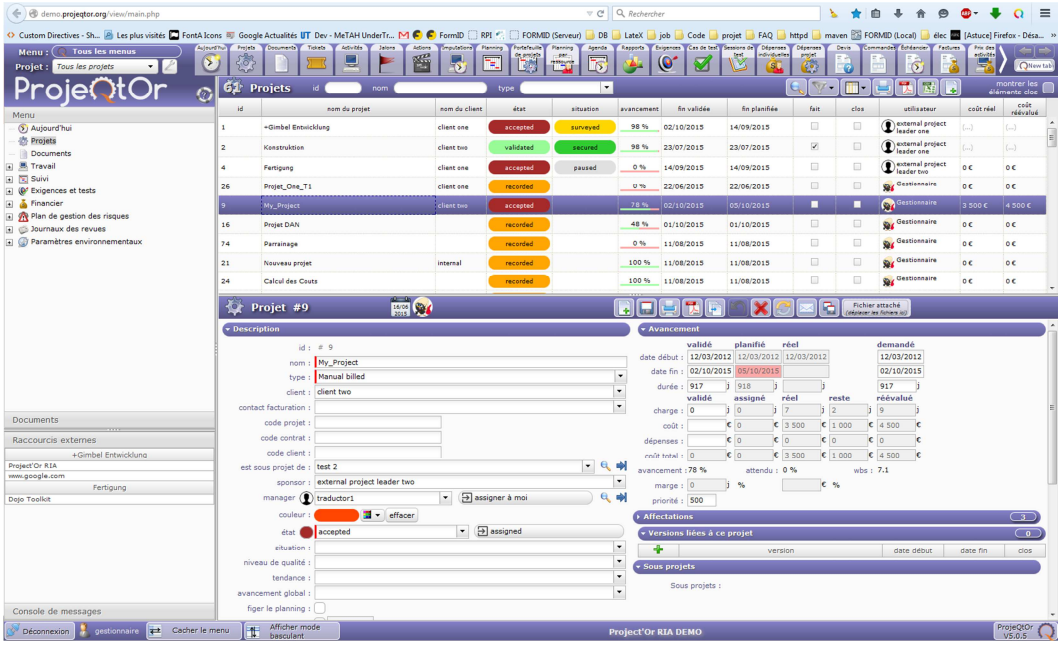
11.1 OPEN WORKBENCH



Source: http://sourceforge.net/projects/openworkbench/, consulté en octobre 2015. Le site officiel ne répond pas.

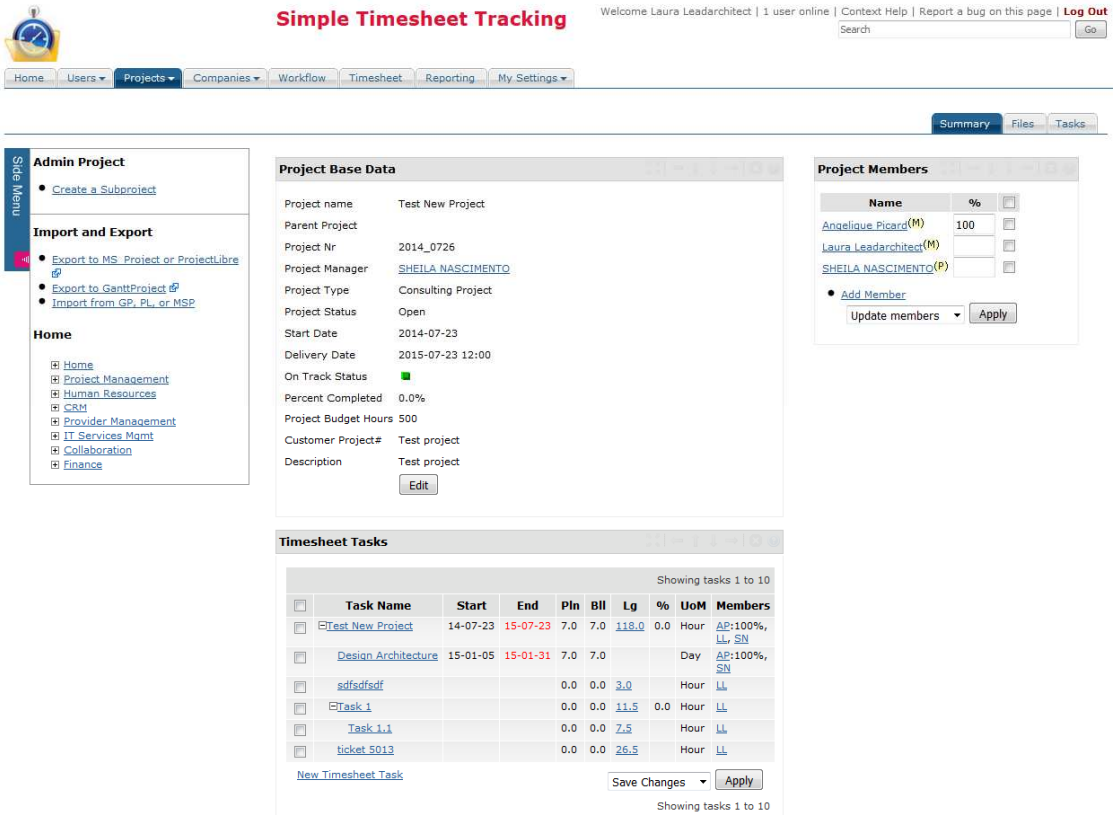
11.2 PROJEQTOR

ProjeQtOr est un logiciel de gestion de portefeuille de projets gratuit.



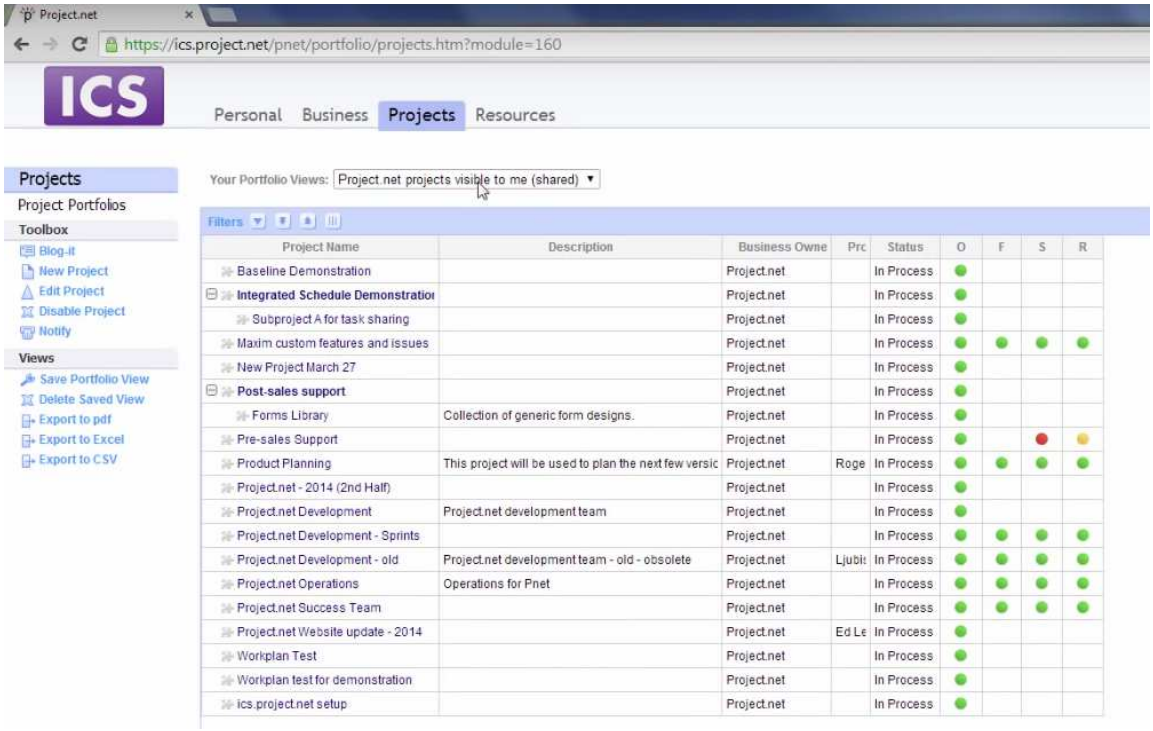
Source : http://www.projeqtor.org/fr/ (consulté en octobre 2015).

11.3 PROJECT OPEN



Source : http://www.project-open.com/index.html (consulté en octobre 2015)

11.4 PROJECT.NET



Source : https://www.project.net/ (consulté en octobre 2015)

11.5 GANTTPROJECT

La copie d'écran de **GanttProject** est visible Figure 49 (page 71).

12 EVÉNEMENTS NOTABLES

Tous ces événements sont représentés sur une frise chronologique, Figure 58 page 82.

Pilotage MOA | Séminaire | Evénement exceptionnel
Réunion | Soutenance de thèse

	Date	Événement
	03/03/2015	Formation fonctionnelle par Viviane Guéraud
	10/03/2015	Réunion avec Viviane Guéraud concernant l'urgence d'avancer sur l'aspect tangible. Lecture ensemble et discussion de chacun des points de la spécification tangible. Nous sommes d'accord pour prendre rendez-vous avec Rémi
	17/03/2015	Préparation de la réunion, puis réunion avec Rémi Pincent, Anne et Viviane pour décider de la suite
	24/03/2015	Réunion avec Anne Lejeune et Viviane Guéraud concernant les priorités.
	27/03/2015	Réunion
	31/03/2015	Travail avec Anne Lejeune sur l'état des lieux de Formid en testant les fonctionnalités pour les prioriser.
	28/04/2015	Journée avec Anne Lejeune pour faire le point et tester
	06/05/2015	Point avec Anne Lejeune
	21/05/2015	Démonstration du fonctionnement tangible à Anne Lejeune et Hamid Chaachoua.
	26/05/2015	Séminaire par Ben-Manson Toussaint concernant un framework d'apprentissage de données perceptivo-gestuelles en chirurgie orthopédique.
	28/05/2015	Réunion avec Nadine Mandran, Vanda Luengo, Jade Thiriat quant au besoin de Jade d'envoyer des actions "tablette" à UT. Prêt des balances à Jade.
	01/06/2015	Tests avec Nadine Mandran et seul pour la partie connexion Formid.imag.fr.
	05/06/2015	Inauguration du bâtiment espace numérique Amiqual4home à Montbonnot.
	16/06/2015	Réunion avec Anne Lejeune et Viviane Guéraud, démo de la tablette de Nathalie Brasset. Et envoi d'informations à Jade Thiriat pour reprendre son appli.
	18/06/2015	Réunion avec Hamid Chaachoua, Yasmina, Viviane Guéraud, Anne Lejeune pour discuter des prérequis techniques et de l'objectif scientifique de la mise à disposition de buchettes et de Formid à partir de septembre dans une école.
	18/06/2015	Réunion avec Jade et Nathalie pour l'application Bouchettes (fin demain pour Jade Thiriat)

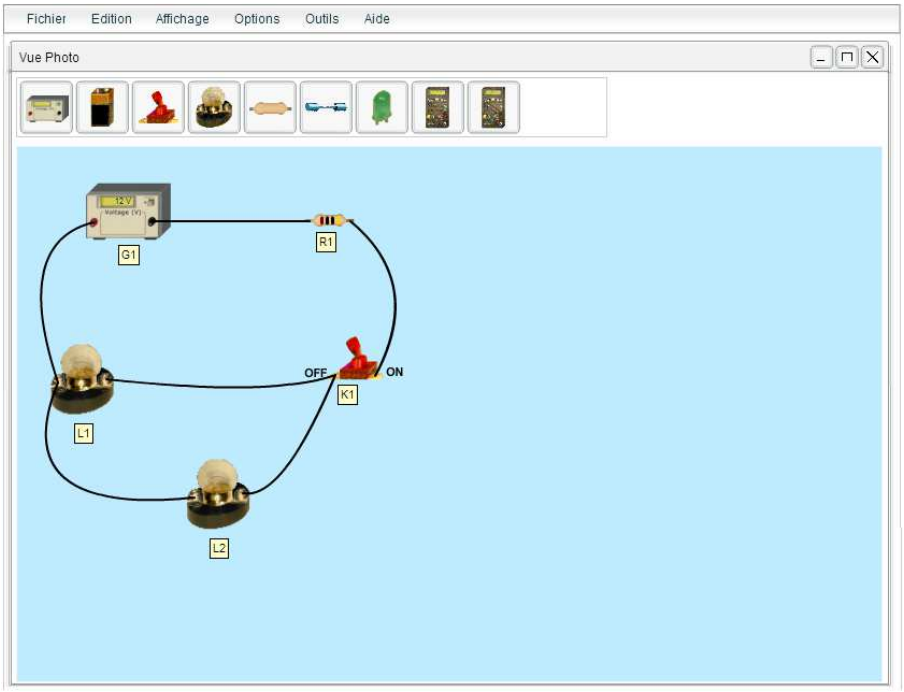
	Date	Événement
	19/06/2015	Réunion avec Cyrille Desmoulins, Anne Lejeune, Viviane Guéraud concernant les praxéologies (didactique) et leur représentation dans OntoPrax, moteur d'ontologies, et prise de notes lors de l'appel de Mohameth François qui a développé le système sur sa machine
	30/06/2015	Séminaire MeTAH de Cyrille Desmoullins concernant Ontoprax
	01/07/2015	Réunion avec Pierre Tchounikine concernant l'intégration de la tablette avec carrés rouges comme simulation de Formid. J'ai proposé que Formid propose un service REST que la tablette appelle quand son état de simulation change.
	03/07/2015	Point complet avec Anne Lejeune et Viviane Guéraud.
	07/07/2015	Séminaire de Catherine Bonnat concernant les praxéologies dans la Biologie de la 6ème à la terminale : levures qui respirent et fermentent
	07/07/2015	Point avec Anne et Viviane concernant mon document de 15 pages "Dialogue avec les simulations"
	15/07/2015	Réunion avec Anne Lejeune pour décider des priorités à venir pour la période estivale.
	31/08/2015	Point avec Anne Lejeune: démonstration des états initiaux/finaux, du dispositif *LettersGame* en mode externe.
	03/09/2015	Point avec Anne Lejeune concernant les contrôles réarmables, le mode élève déporté (test ok), les variables de scénario et le mode externe normal. Antivirus désactivé pour les problèmes de WebSockets.
	08/09/2015	Mini point avec Anne : j'ai montré les listes du dispositif externe (users, exercises), et elle m'a fait part de son désir d'avoir un serveur d'ontologies "simulé".
	11/09/2015	Réunion improvisée avec Pierre Tchounikine pour SimuBûchettes V1 et V2 et le lien à FORMID sans modification de la simulation
	15/09/2015	Point complet avec Anne Lejeune et Viviane Guéraud sur l'état du projet et les priorités :7 priorités, la première étant la nécessité d'un fac-similé de praxéologies, la dernière étant de chercher à se passer des WebSockets.
	16/09/2015	Réunion avec Cyrille Desmoulins pour le serveur Virtuoso qui ne fonctionne pas : rappel et état des lieux.
	17/09/2015	Rendez-vous avec M. Fayolle pour faire le point sur le stage.
	21/09/2015	Point rapide avec Anne Lejeune.
	06/10/2015	Point avec Anne Lejeune concernant toutes les petites idées d'évolution.
	12/10/2015	Soutenance de thèse de Ben-Manson Toussaint.
	13/10/2015	Point complet avec Anne Lejeune et Viviane Guéraud : 6 points sur le module tuteur et un pour le fac-similé de praxéologies : données réelles à renseigner.
	15/10/2015	Soutenance de thèse de Reinaldo Javier Saavedra Gomez.
	21/10/2015	Point avec Anne Lejeune et Viviane Guéraud concernant l'affectation d'une erreur courante à un contrôle.

	Date	**Événement**
	10/11/2015	Réunion avec Anne Lejeune et Viviane Guéraud concernant les praxéologies, pour décider du développement d'un serveur de praxéologies.
	10/11/2015	Séminaire de Nathalie Brasset : "*Etude de facteurs liés à l'histoire didactique pour la conception d'un modèle de décisions didactiques de l'enseignant dans un EIAH*"
	26/11/2015	Préparation de l'expérimentation en classe.
	26/11/2015	Expérimentation en classe de CE1.

13 PHOTOS DES DIFFÉRENTS DISPOSITIFS EXTERNES

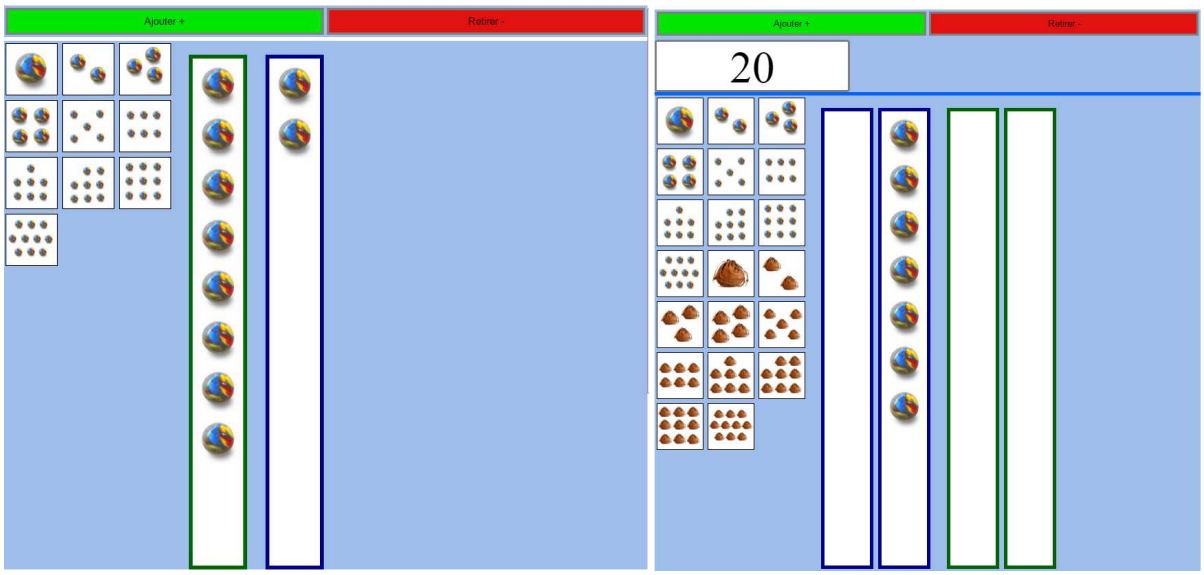
Cette partie montre différents exemples de dispositifs externes connectables à FORMID.

13.1 LE MICROMONDE ÉLECTRIQUE *TPELEC*



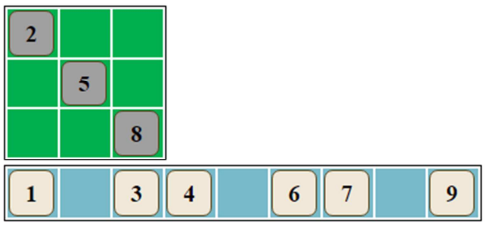
13.2 LA NUMÉRATION AVEC DES BILLES *TPNUM*

Cette simulation se décline en trois versions: complément à 10 (à gauche), à 100 ou à "n" (à droite).

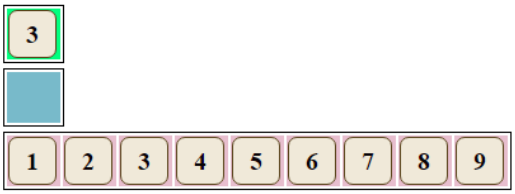


13.3 LA MANIPULATION DE LETTRES *LETTERSGAME*

Pour réaliser un carré magique :

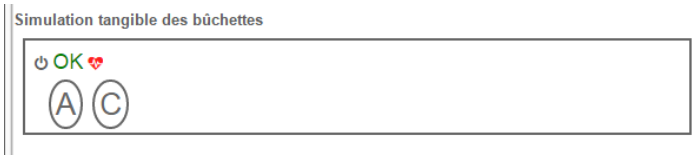


Pour travailler le complément à 10 :



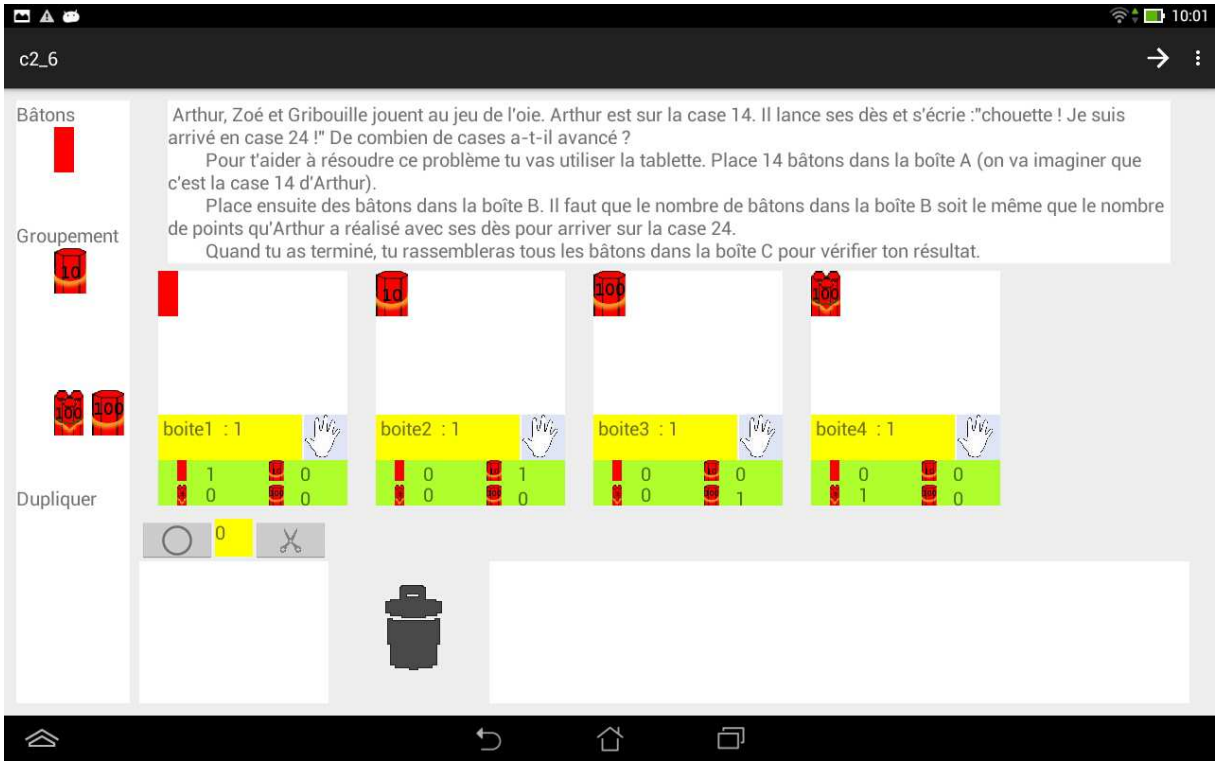
13.4 LA MANIPULATION TANGIBLE *BÛCHETTES*

Partie visible à l'écran : le battement cardiaque indique la connexion fonctionne, les lettres A et C représentent les boîtes détectées.



Partie tangible :

13.5 LA SIMULATION SUR TABLETTE TACTILE *SIMUBÛCHETTES*



LISTE DES FIGURES

LISTE DES TABLEAUX

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHONE-ALPES

Extensions de la suite logicielle FORMID pour assister la conception et le suivi de situations pédagogiques exploitant des simulations ou des objets tangibles. Mémoire d'ingénieur, 2016.

Emmanuel LÉTONDOR Grenoble, 2016

RÉSUMÉ

Ce mémoire présente mon intervention sur la suite logicielle FORMID (Formation Interactive à Distance), une plateforme permettant de travailler sur des situations pédagogiques particulières, créée par l'équipe MeTAH du Laboratoire d'Informatique de Grenoble (LIG). Ce logiciel permet à un auteur de concevoir des scénarios pédagogiques autour d'une simulation ou d'un micromonde. L'élève peut ainsi réaliser les exercices conçus, en manipulant des dispositifs externes, sous la supervision d'un tuteur. Ce document décrit les améliorations techniques et fonctionnelles que j'ai apportées à FORMID. Une attention toute particulière est portée aux dispositifs externes dont la gestion a beaucoup évolué, avec essentiellement la connexion à un dispositif tangible et à une simulation sur tablette tactile. Il sera également expliqué pourquoi l'assistance à l'auteur grâce à des données praxéologiques n'a pas pu progresser comme attendu. Finalement, je présenterai les aspects théoriques de l'ingénierie logicielle rencontrés et j'exposerai mon parcours dans la gestion du projet.

Mots-clés : Formation, Apprentissage, Didactique, Praxéologie, Simulation, Micromonde, Scénario pédagogique

ABSTRACT

This report is about the work I did on the FORMID (Interactive remote learning) software, a platform allowing to work on particular learning situations, made by MeTAH team within Computing Laboratory of Grenoble (LIG). This software allows an author to build teaching scenarios around simulations and microworlds. A student is then able to perform an exercise by manipulating external devices, under the supervision of a tutor. This paper describes technical and functional improvements I've made in FORMID. Particular attention is given to external devices whose management has greatly evolved, mainly with connection to a tangible device and to a tactile tablet simulation. It will be also explained why author assistance provided with praxeological data couldn't have progressed as planned. Eventually, I'll show theoretical aspects of software engineering I've met, and I will outline my progression in project management.

Keywords: Learning, Teaching, Didactic, Praxeology, Simulation, Microworld, Scenario