IUT Nancy Charlemagne

Université de Lorraine

2 Ter Boulevard Charlemagne

54052 Nancy Cedex

Département Informatique

Portage web d'un exerciseur de programmation

Rapport de stage de DUT Informatique

Laboratoire Lorrain de Recherche en Informatique et ses Applications

Théodore LAMBOLEZ

47e Promotion 2013/2015

IUT Nancy Charlemagne

Université de Lorraine

2 Ter Boulevard Charlemagne

54052 Nancy Cedex

Département Informatique

Portage web d'un exerciseur de programmation

Rapport de stage de DUT Informatique

Laboration Lorrain de Recherche en Informatique et ses Applications

615 Rue du Jardin Botanique

54600 Villers-lès-Nancy

Etudiant : Théodore LAMBOLEZ

Tuteurs : Gérald OSTER & Martin QUINSON

Marraine du stage : Isabelle DEBLED-RENNESSON

Remerciements

Je tiens tout d'abord à remercier l'ensemble des enseignants de l'IUT Nancy Charlemagne qui m'a formé et préparé à accomplir avec succès mon premier stage professionnel en informatique.

Premièrement, je remercie **Madame Bellalem, enseignante à l'IUT Charlemagne de Nancy** pour m'avoir guidé dans mes recherches et conseillé de choisir ce sujet de stage.

Ensuite, je suis reconnaissant envers **Monsieur Oster et Monsieur Quinson, enseignant-chercheurs au LORIA**, qui m'ont accepté, encadré et fourni de nouvelles méthodes de travail au cours de mon stage.

Un grand merci à **Matthieu Nicolas, Ingénieur Jeune Diplômé, embauché au LORIA**, qui m'a accompagné, renseigné et aidé dans ma progression.

J'exprime toute ma gratitude à **Madame Debled-Rennesson** qui est passé me voir et m'a aiguillé sur le plan de ce rapport.

Enfin, merci à **mes camarades Baptiste Mounier, Benjamin Thirion et à un doctorant Luc André** pour avoir participé à cette incroyable expérience dans la bonne humeur.

Tables des matières

I.Introduction.....4
II.La Programmer's Learning Machine.....5
1.Présentation de l'entreprise :.....5
2.Mission pédagogique du LORIA.....6
3.La PLM version web.....7
III. Portage de mondes.....19
1.Positionnement du sujet.....19
2.Besoins et organisation.....19
3.Conception et Réalisation.....25
IV.Conclusion.....31
Bibliographie.....32
Glossaire.....33
Table des annexes.....34
Annexe 1 : Capture d'écran de mon reporting.....35
Annexe 2 : Diagramme de Gantt.....36

I. Introduction

Dans le cadre de mon DUT Informatique à l'Université de Lorraine, je désirais découvrir le monde de la recherche scientifique dans ce domaine. J'ai réussi à intégrer le Laboratoire Lorrain de Recherche en Informatique et ses Applications afin de participer à un projet pédagogique.

Chaque année, environ 100 étudiants de Classes Préparatoires arrivent à TELECOM Nancy pour devenir ingénieurs en Informatique. Au cours de leurs deux années intensives, ils n'ont appris que quelques bases de programmation. L'école a donc pour mission de leur enseigner la théorie mais doit aussi les accompagner dans la pratique. Certains de ses enseignants sont chercheurs au LORIA et ont développé au cours de leur recherche la Programmer's Learning Machine. Cette application propose des exercices attractifs sur des thèmes classiques de l'art de la programmation comme les tris ou la récursivité.

Aujourd'hui, la PLM a atteint ses objectifs mais avec la croissance du nombre de personnes désirant travailler dans l'informatique de nouveaux besoins pédagogiques sont apparus. Les enseignants-chercheurs à l'origine du projet ont décidé de produire une nouvelle version web permettant de répondre à ces nouvelles attentes.

Ma mission a consisté à intégrer les mondes de l'ancien client lourd Java dans le nouveau service web.

Dans un premier temps, je vous présenterai la PLM en passant par le LORIA, ses objectifs pédagogiques et la nouvelle version. Puis, je vous parlerai de ma mission en précisant sa place dans le développement en cours, mon organisation et mes réalisations.

II. La Programmer's Learning Machine

1. Présentation de l'entreprise :

Le Laboratoire Lorrain de Recherche en Informatique et ses Applications dit LORIA est situé entre Vandoeuvre et Villers-lès-Nancy. C'est en partenariat entre le Centre National de Recherche Scientifique, l'Institut National de Recherche en Informatique et en Automatique et l'Université de Lorraine. Il étudie les sciences et technologies de l'information et de la communication, forme des étudiants en doctorat par le biais de thèses et réalise des transferts de technologie. Ces transferts sont des accords entre le laboratoire et des entreprises. Les recherches sont à l'échelle internationale (48 nationalités sont présentes) et souvent en collaboration avec d'autres laboratoires. Le LORIA a participé notamment au développement de plusieurs logiciels comme SmartEiffel, Tom ou MPFR.

Afin d'être opérationnel, le LORIA embauche 14 personnes pour l'administration (chiffres fin 2013). Il a un rôle essentiel dans l'éducation d'où son partenariat avec l'Université de Lorraine et reçoit 102 doctorants, 23 post-doctorants, 4 Attachés Temporaire d'Enseignement et de Recherche, 2 chercheurs contractuels et 27 ingénieurs scientifiques. Dans le but de conserver une organisation optimale, le LORIA dispose de plusieurs services coordonnés : la direction, le service d'informatique commun, de gestion, de Communication et de missions. Ce dernier porte sur trois thèmes : la CyberSanté, la Sécurité et les Systèmes cyber-physiques. Quant aux recherches, elles sont réparties en 30 équipes regroupées en 5 départements :

- Algorithmique, calcul, image et géométrie
- Méthodes formelles
- Réseaux, systèmes et services
- Traitement automatique des langues et des connaissances
- Systèmes complexes et Intelligence artificielle

Cependant, il n'est pas impossible que plusieurs membres d'équipes différentes travaillent sur un projet commun. Par exemple, Gérald OSTER fait partie de l'équipe COAST et Martin QUINSON de l' équipe Veridis. Généralement, ces projets portent sur une mission spécifique au LORIA comme l'éducation, ce qui confirme sa relation avec l'Université de Lorraine.

2. Mission pédagogique du LORIA

Le LORIA accueille 174 chercheurs dont 114 enseignants dans toute la Lorraine, dans plusieurs établissements comme l'IUT Charlemagne ou même TELECOM Nancy. Il est donc logique que le laboratoire mette en place un projet pédagogique. Depuis 2008, Gérald Oster et Martin Quinson, deux enseignants-chercheurs à TELECOM Nancy travaillent sur un logiciel éducatif ayant pour but d'apprendre la programmation. Ce domaine a une base théorique pouvant être donnée en cours, dans des livres ou sur Internet mais sa pratique est essentielle. « C'est en forgeant que l'on devient forgeron » ! L'objectif de l'application est donc d'apporter le plus d'expérience possible à l'étudiant de manière attractive. La base de l'écriture de programme est l'algorithmique, c'est pourquoi, l'élève doit s'entrainer à réfléchir et répondre à des problèmes divers.

En outre, le programme peut être utilisé par tous, apprenants et enseignants. Il est donc nécessaire que l'exerciseur propose un mode « édition » permettant la création de nouveaux exercices, ou même la modification de ceux-ci. La programmation est un domaine subtil et un certain changement peut apporter une réelle difficulté supplémentaire. Grâce à Internet, les étudiants peuvent trouver des éléments de réponses faces à des problèmes classiques en informatique. Avec cet outil, le professeur peut contrer la routine avec une originalité obligeant les étudiants à réfléchir.

En supplément, si l'enseignant offre un nouveau cours, il peut facilement ajouter une série d'exercices sur le thème abordé. Cependant, la communication est une pièce maitresse de l'apprentissage. Ainsi les étudiants et les utilisateurs en difficulté ou curieux ont la capacité d'appeler le pédagogue ou de lui envoyer une question. Cela permet de conserver une formation suivie.

Cette année, un sondage a été donné aux élèves et enseignants afin qu'ils expriment leur avis sur le logiciel. Il est globalement positif et prouve que le programme est un outil efficace. L'algorithmique peut être considérée comme la base théorique de la programmation mais la pratique s'étend sur plusieurs langages. C'est pourquoi, l'application propose de développer en quatre langages de programmation. Le LORIA et les écoles d'ingénieurs ont un lien fort avec l'international, en conséquence la plateforme d'apprentissage est disponible aussi en anglais, en italien et en portugais (brésilien).

Malgré l'étendue des ordinateurs fonctionnant sous Windows, les informaticiens, au contraire d'utilisateurs lambda, peuvent s'adapter rapidement à d'autres systèmes d'exploitation. Afin de satisfaire tout le monde, l'exerciseur est disponible aussi sur Mac Os et Linux. Comme souvent dans la recherche et l'apprentissage de l'informatique, les valeurs de l'open-source sont respectées.

En effet, le software est sous-licence GPLv3 ce qui rend sa distribution simple et gratuite. Il est actuellement hébergé sur GitHub (code source, wiki et gestionnaire d'anomalies) et est donc librement modifiable. Aujourd'hui, ce programme est à sa dernière version Java et est appelé la Programmer's Learning Machine ou aussi PLM. Il est disponible à cette adresse : http://www.LORIA.fr/~quinson/Teaching/PLM/. Toutefois, le projet peut être encore plus développé et il a été décidé de réaliser une nouvelle version web permettant de répondre à de nouveaux objectifs.

3. La PLM version web

a) Nouveaux objectifs

La Programmer's Learning Machine Java commençait à devenir vieillissante et devait être remise au goût du jour. Elle vise plusieurs utilisateurs (enseignants, élèves, chercheurs) et il est donc logique que la nouvelle permette que chacun apporte des nouveautés par le biais du travail collaboratif. Afin qu'elle reste fonctionnelle et stable, l'intégration continue est cruciale et **Travis CI*** a été choisi pour réaliser des tests à chaque ajout au dépôt GIT. Afin que les utilisateurs mais aussi les contributeurs puissent s'exprimer, il est important de faciliter leur communication soit par messages privés ou avec un forum. De plus, il semble évident que l'application autorise des commentaires à propos des exercices. Les professeurs peuvent avoir des exercices précis qui ne seraient pas disponibles sur le web PLM. Un nouvel éditeur d'exercice semble obligatoire. Il devrait à la fois donner la possibilité d'ajouter des activités dans les mondes existants mais aussi de nouveaux univers. Les enseignants doivent garder un suivi de leur classe, c'est pourquoi la PLM a l'obligation de pouvoir stocker de manière efficace le travail des étudiants mais aussi leur activité (temps de réflexion, changement d'exercices). Ensuite, il est nécessaire d'avoir un système d'authentification afin de pouvoir trier les données par profil. Pour mieux utiliser ces données, un système de filtres et de statistiques sur une classe par exemple serait judicieux.

Avec ces informations, l'enseignant peut alors compliquer un exercice rapidement réussi par la majorité des étudiants ou l'inverse en cas de blocage fréquent. Pour les problèmes récurrents, il pourrait alors afficher un message d'aide précis. Le plagiat est une plaie dans l'apprentissage de la programmation et il serait fort utile d'implémenter un détecteur comparant les travaux des étudiants.

Du côté étudiants, les exercices doivent être attractifs et c'est déjà le cas avec le programme Java. Afin d'améliorer encore ce point, l'interface doit être embelli et un système de badges (récompenses) pourrait être mis en place afin de renforcer les motivations des apprenants.

Pour mieux comprendre l'exécution de leur programme, un **debugger*** serait intéressant. Pour encore améliorer la force d'apprentissage de l'application, l'ajout de nouveaux langages sera toujours un plus. Et pour certains exercices plus complexes, la possibilité de développer en binôme est une bonne idée.

Enfin, chaque utilisateur potentiel a des outils préférentiels, par exemple Moodle pour l'apprentissage. Il serait donc utile que la web PLM puisse être insérée dans une infrastructure de type e-learning. Afin de réaliser ces objectifs et de conserver une structure **MVC***, une architecture 3-tiers a été choisie.

b) Architecture 3-tiers

L'architecture 3-tiers (client, serveur d'application, serveur de données) permet de créer un service web avec stockage et utilisation de données. Le serveur d'application est l'ancien programme Java PLM couplé au framework Play afin de recycler aux mieux le travail déjà effectué. Il reçoit le code de l'étudiant, le compile et l'exécute. Pendant ce temps-là il va envoyer des opérations au client. Ce dernier est une application AngularJS exécutée par le navigateur. Celui-ci va recevoir des requêtes HTTP contenant les opérations envoyées par une **webSocket***. Elles vont être traitées par le service connexion de l'application qui va en extraire et expédier au contrôleur les informations nécessaires pour déterminer quelle tâche effectuer. Ce dernier va modifier le modèle et actualiser la vue en conséquence.

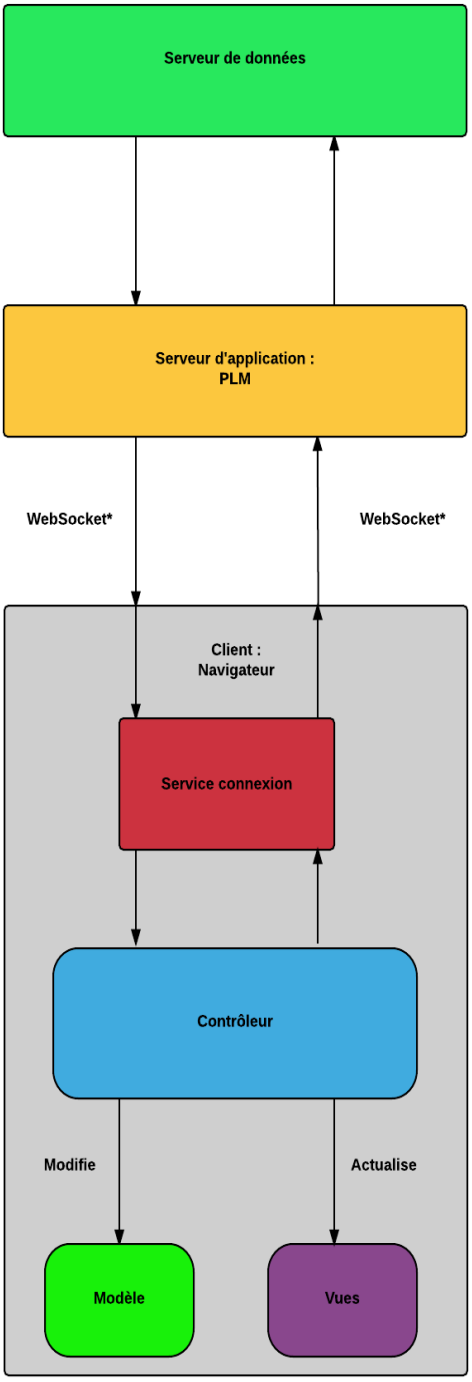


Illustration 1: Schéma de l'architecture de webPLM.

c) **Affichage d'accueil**

La Programmer's Learning Machine est une application permettant aux nouveaux étudiants de Telecom Nancy venant de Classes Préparatoires d'apprendre la programmation. Elle propose plusieurs exercices répartis dans différents mondes.

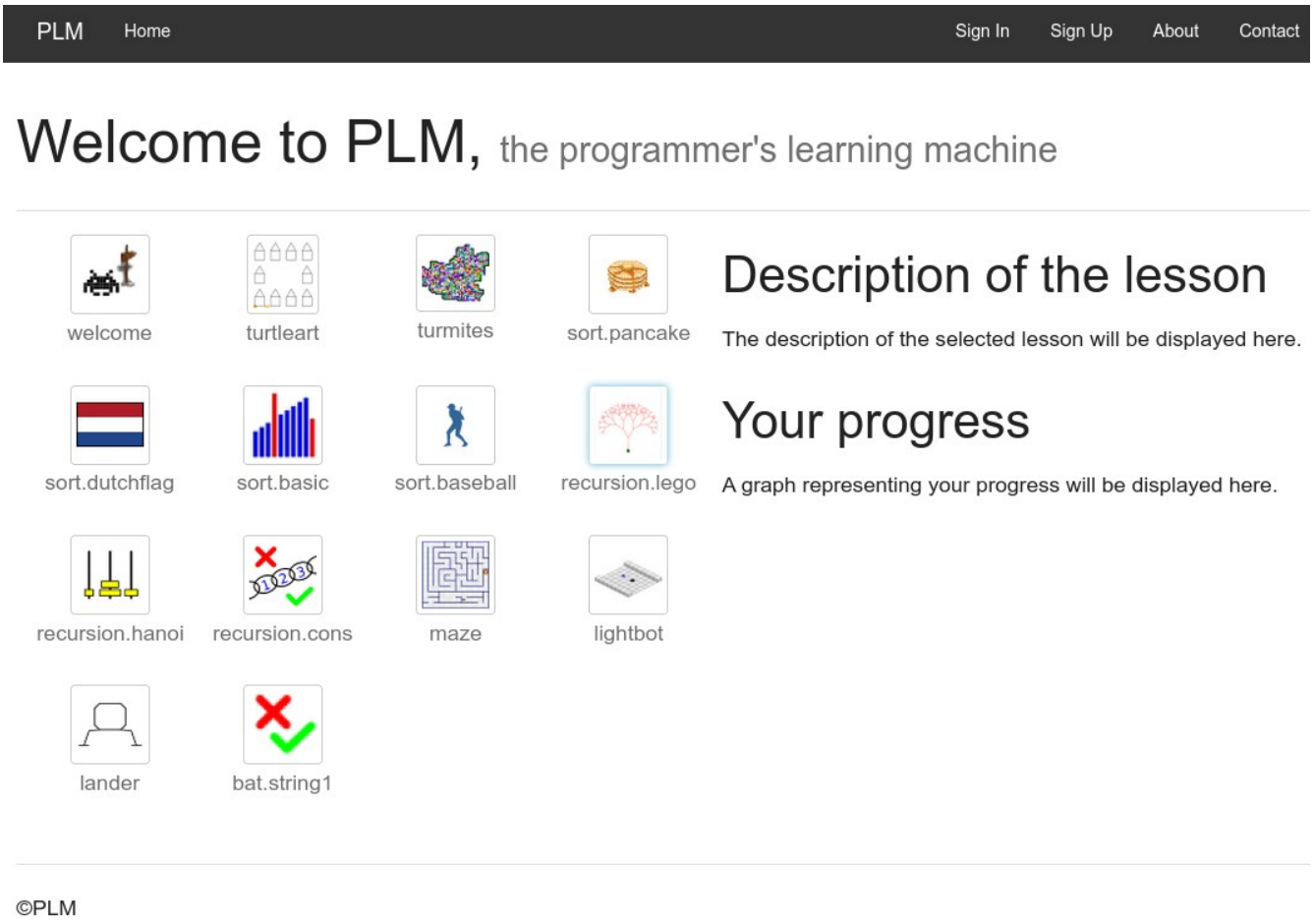


Illustration 2: Page d'accueil de la PLM.

La page d'accueil de la PLM est composée comme toutes les autres d'une bande noire supérieure où on pourra trouver un bouton Home permettant de revenir à celle-ci. Les boutons à droite permettent respectivement de s'inscrire, de se connecter, de retrouver des informations et de contacter l'administrateur. Comme prévu, l'utilisateur peut sélectionner son monde parmi plusieurs illustrés par une image et un titre. A droite, il peut voir la description de la leçon du monde et son avancement dans celui-ci.

d) Affichage des mondes

Chaque monde à une fenêtre avec les mêmes boutons principaux :

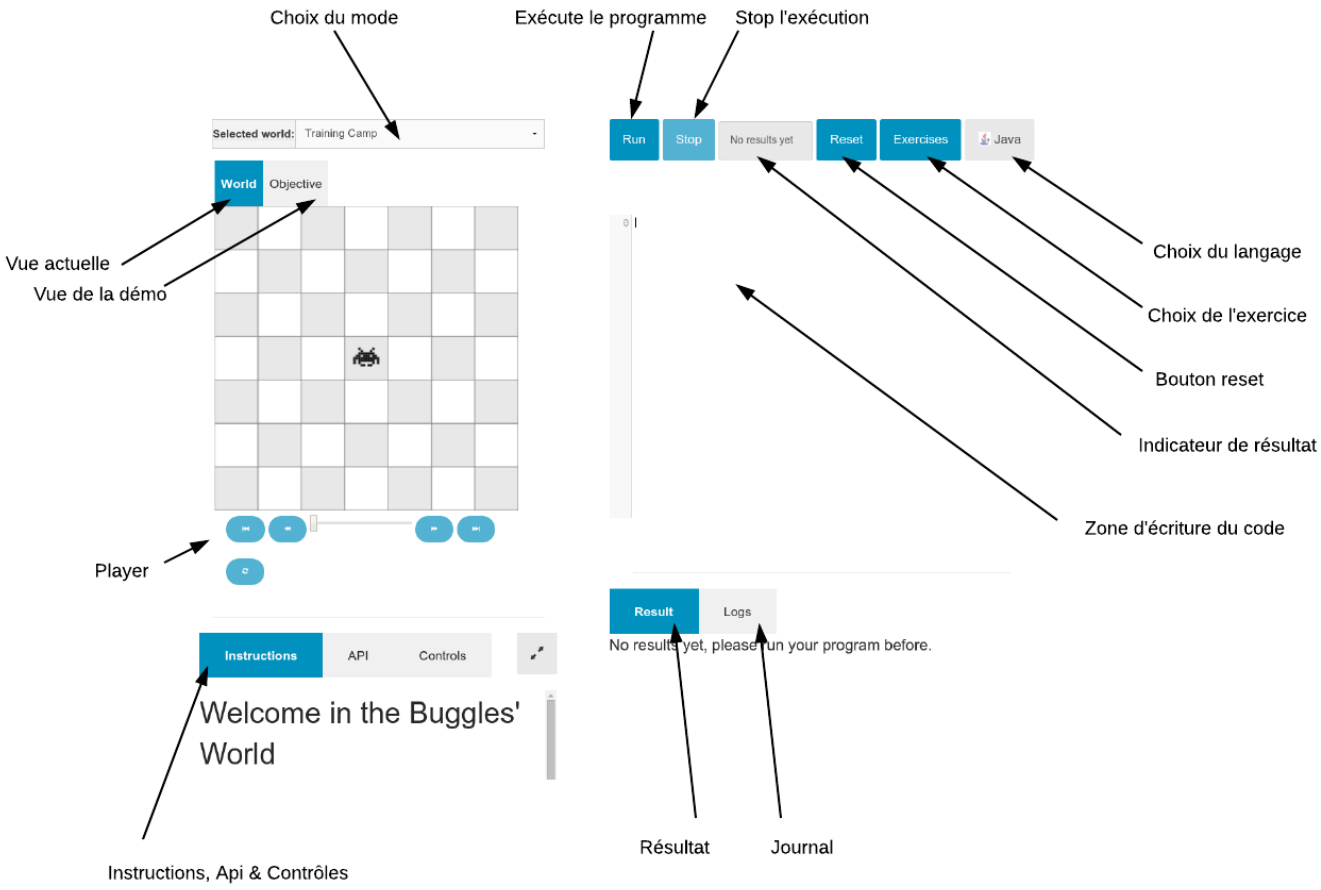


Illustration 3: Page d'un monde.

Chaque monde propose un ou plusieurs exercices de niveaux de difficulté différents. En effet, il porte sur des algorithmes connus résolvant des problèmes courants comme les tris ou la récursivité. Les exercices sont sélectionnés grâce à un bouton ouvrant une fenêtre à choix et peuvent être effectués en python, java, en scala grâce à un bouton. Cela permet de familiariser l'apprenti avec plusieurs langages plus ou moins stricts.

En effet, chaque monde possède sa propre vue représentant l'état actuel de l'exercice. Chaque exercice dispose aussi d'un ou plusieurs mondes où les données vont changer. Par conséquent, on peut tester sa solution avec différents nombres d'éléments. Par exemple, dans le monde des tris, il existe une version avec 10 valeurs à trier et une autre avec 150. On peut choisir le nombre voulu avec une liste déroulante.

L'élève dispose naturellement d'une zone d'écriture de son code et il peut l'exécuter ou l'arrêter. Un encadré indique si des résultats sont disponibles, si l'exercice est réussi. Dans le cas contraire, il nous indique d'aller consulter une zone d'information située en bas à gauche, accompagné des **logs***. Le player permet d'accéder directement à la dernière étape du programme ou de revenir à la première. Grâce à lui, on peut aussi avancer ou reculer étape par étape. Cet outil est aussi disponible pour la démonstration. En effet, celle-ci affiche les différentes étapes du programme correct.

L'étudiant dispose des consignes lui expliquant le but de l'exercice et le principe de l'algorithme à utiliser dans l'onglet instructions, un ou plusieurs indices sont disponibles en cliquant sur des liens.

A côté se trouve l'onglet API, c'est-à-dire de la liste des méthodes utilisables avec leurs paramètres, leur effet et leur type de retour. Enfin, le dernier dévoile un curseur permettant de sélectionner la vitesse d'affichage désirée.

Chaque monde dispose d'une ou plusieurs vues (sauf ceux utilisant des strings ou des booléens) afin de bien représenter le résultat de l'algorithme écrit par l'apprenti. Les vues aident l'élève à bien comprendre ce que produit son code et par conséquent à corriger ses éventuelles erreurs. Les secondes vues permettent de mieux visualiser les changements effectués au cours du temps. Ces affichages ont logiquement un lien précis avec les valeurs du monde utilisées par l'étudiant :

- **Le monde des Buggles :**

Le monde des Buggles est composé d'une grille (cellules), de murs, de différents objets et d'un ou plusieurs Buggles. Généralement, l'exercice consiste à écrire un programme qui déplace le buggle en respectant des règles jusqu'à une certaine case (ici jaune). Dans certains exercices, il faut même contrôler plusieurs Buggles à la fois ou aller chercher plus d'un objet.

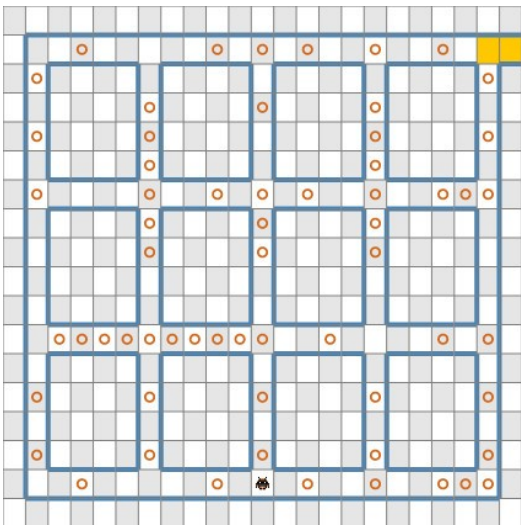


Illustration 4: Le monde des Buggles

- **L'art des tortues :**

L'art des tortues est une référence en programmation, j'avais dû déjà le réaliser en cours d'Interface Homme-Machine. Le but de l'exercice est de déplacer la tortue en décidant si elle doit tracer ou non afin de dessiner les différentes figures. On peut apercevoir une mesure en jaune qui nous informe de la taille des segments. Grâce à la possibilité de tourner la tortue (en choisissant l'angle voulu), on peut aussi tracer des cercles. Dans certains exercices, on doit aussi changer la couleur du trait au bon moment.

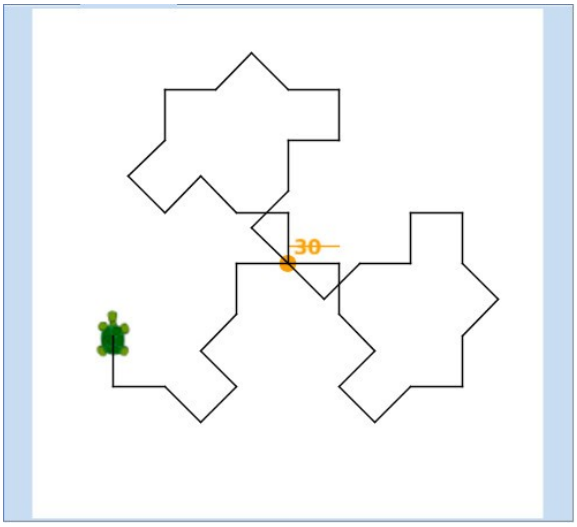
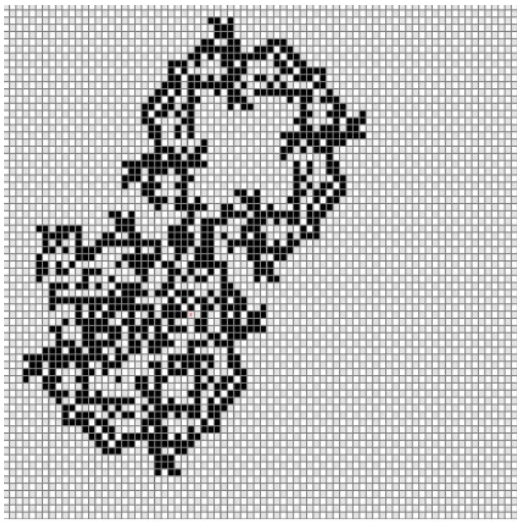


Illustration 5: Le monde des tortues.

- **Le monde des Turmites :**

Le monde des turmites présente l'automate « fourmi de Langton » et des variantes à celui-ci. Il est représenté par une grille où chaque case peut-être noire ou blanche. Le programme va simuler le déplacement d'une fourmi en coloriant ou décolorant des cases au fur et à mesure de son passage. Les exercices proposent différentes règles avec par exemple, des turmites multi-colores.

Illustration 6: Le monde des turmites.



- **Le monde des pancakes :**

Le monde des pancakes représente une pile de pancakes qu'il faut trier en fonction de leur taille. Le plus grand doit être en dessous et ainsi de suite afin de former une pyramide. Ils sont illustrés par des rectangles jaunes. Si deux pancakes ont une différence de taille de un, alors un rond rose se dessine entre eux. Sachant que le tri s'effectue en retournant un ou plusieurs pancakes, on dessine une spatule afin d'indiquer où s'est trouvé le dernier inversement. Enfin dans certains exercices, on ajoute une difficulté supplémentaire, en effet, la face brûlée doit toujours être vers le plateau. De plus, on affiche le nombre de retournements réalisés en haut à droite.

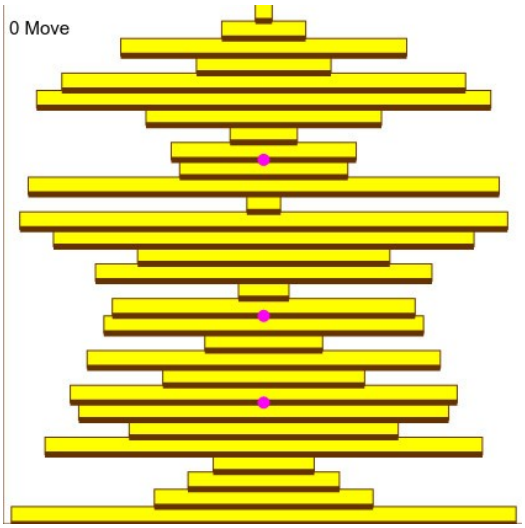


Illustration 7: Le monde des pancakes.

- **Le monde du drapeau hollandais :**

Le monde est composé de différentes bandes de couleurs qu'il faut trier afin de former le drapeau hollandais (bleu, blanc, rouge). Comme dans le monde des pancakes, on écrit le nombre de déplacements effectués. Dans certains exercices, il n'y a que deux couleurs à trier. Afin que l'utilisateur soit sûr que son algorithme soit correct, il y a même une version avec 300 lignes !

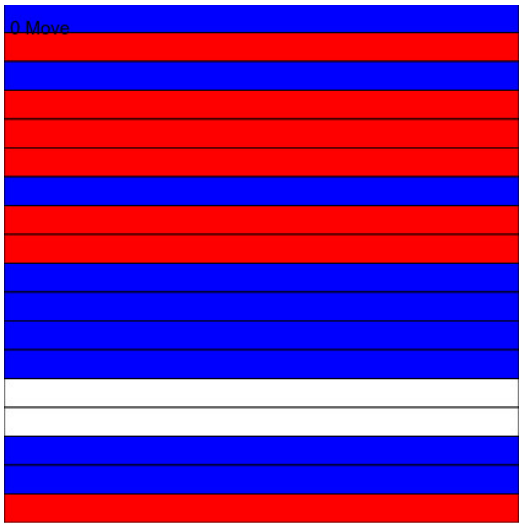
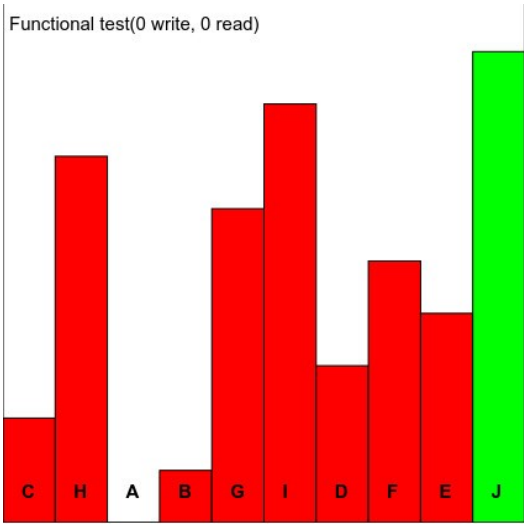


Illustration 8: Le monde du drapeau hollandais..

- **Le monde des tris :**

Le monde des tris est constitué d'un diagramme en barres, où chaque colonne vaut une valeur du tableau. Il faut trier le tableau dans un ordre croissant, si une barre est bien placée, elle change du rouge au vert. La différence entre les exercices de ce monde est l'algorithme utilisé. On notera aussi l'affichage du nombre de lectures et d'écritures faits pendant l'exécution du code.

Illustration 9: Le monde des tris.



- **Le monde baseball :**

Le monde baseball est constitué de bases où se trouvent des buggles. Chaque élément a une couleur et l'objectif est de placer tous les buggles dans leur base respective. Un cercle entoure les positions, s'il est rouge, le buggle est mal placé sinon il devient gris. Il y a toujours une position vide afin de permettre le déplacement. Les buggles ne peuvent se déplacer que dans une position de leur base actuelle ou voisine. Une flèche pointillée indique le mouvement du buggle, elle est de la même couleur que celui-ci. Le nombre de mouvements est aussi inscrit en haut à droite de l'affichage.

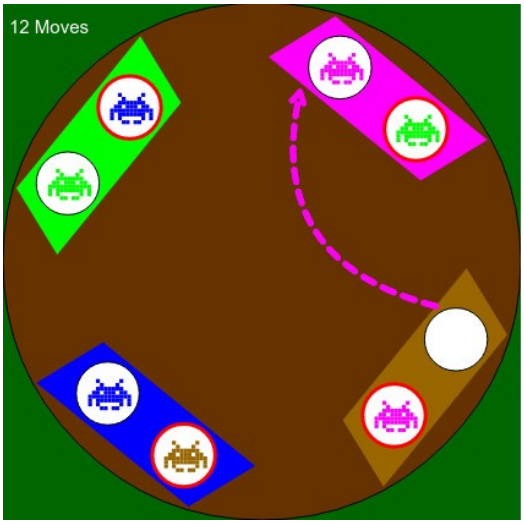
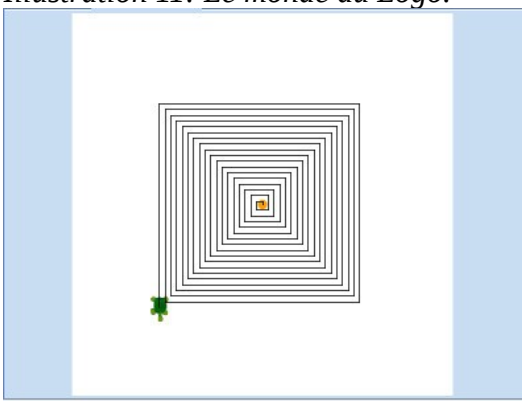


Illustration 10: Le monde baseball.

- **Le monde du Logo :**

Le monde du Logo est similaire à celui de l'art des tortues mais les exercices portent sur des algorithmes récursifs. On peut toujours diposer d'une mesure de segment.

Illustration 11: Le monde du Logo.



- **Le monde des tours d'Hanoï :**

Le monde des tours d'Hanoï est aussi un exemple connu de la programmation. Un algorithme récursif permet de réussir l'objectif qui consiste à déplacer tous les éléments d'une tour vers une autre tout en conservant leur ordre décroissant (le plus grand est tout en bas). On dispose d'une tour supplémentaire pour accomplir la mission car il est interdit de placer un élément sur un autre inférieur. On écrit aussi le nombre de mouvements effectués en haut à droite.

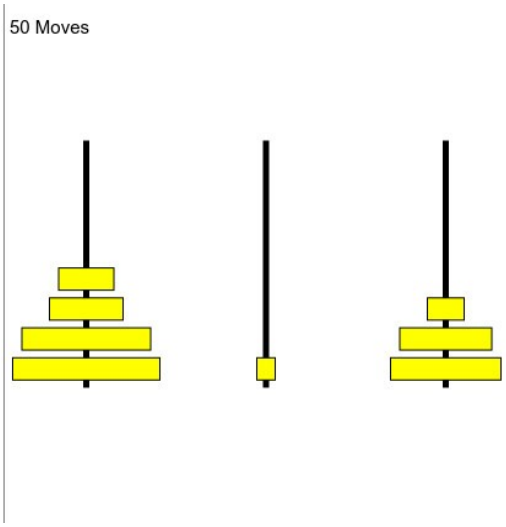


Illustration 12: Le monde des tours d'Hanoï.

- **Monde des listes et des chaînes de caractères :**

Ceux-ci n'ont pas de réelles vues et de démonstrations. Ils sont affichés en texte. Le monde des listes porte sur des algorithmes récursifs, des tests simples de fonctions et le suivant sur la manipulation des chaînes de caractères.

```
length( [1, 2, 3] )=null
length( [1, 1, 1] )=null
length( [1, 2, 1, 3] )=null
```

Illustration 13: Mondes textuels.

- **Monde Labyrinthe :**

Il utilise la même vue que celui des Buggles car l'objectif consiste à déplacer un Buggle dans un Labyrinthe jusqu'à un objet. L'algorithme doit pivoter le Buggle pour éviter les murs et le faire avancer.

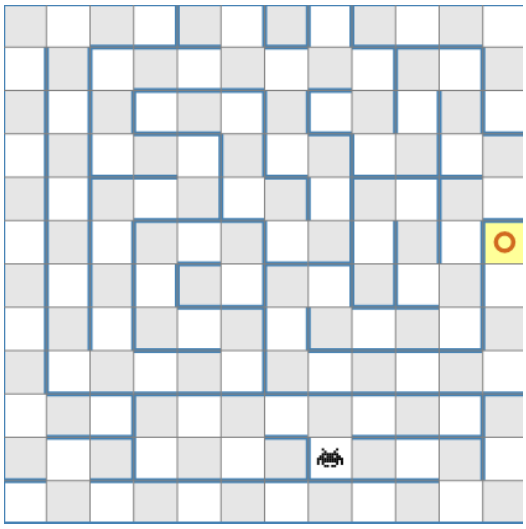


Illustration 14: Le monde Labyrinthe.

- **Le monde du Lightbot :**

Le monde du Lightbot dispose d'une vue 3D (il est possible de pivoter l'affichage). Le but consiste à déplacer la flèche représentant le robot et d'allumer les ronds noirs.

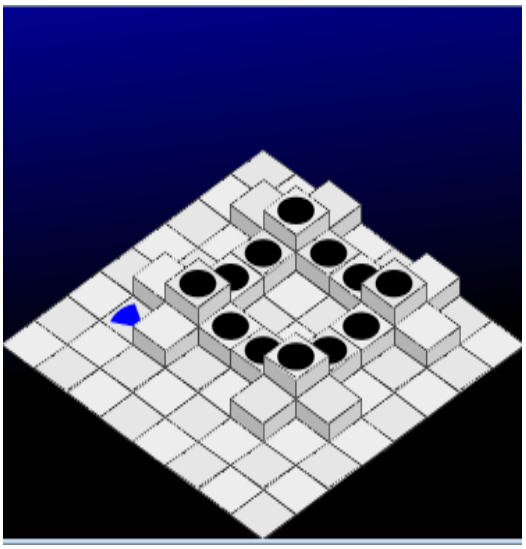


Illustration 15: Le monde du Lightbot.

- **Lunar lander :**

Ce monde consiste à faire atterrir un vaisseau sans l'endommager. Les exercices portent sur la physique car notre programme doit prendre en compte la position du module, la gravité, le niveau de carburant et même la poussée du moteur. On pourra aussi retrouver ces informations en haut à droite et gauche de l'affichage.

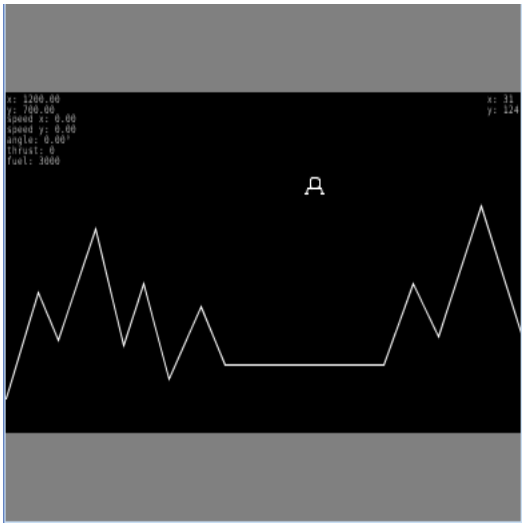


Illustration 16: Le monde du Lander.

III. Portage de mondes

1. Positionnement du sujet

La nouvelle Web PLM était déjà en cours de développement lorsque je suis arrivé. Matthieu Nicolas, un Ingénieur Jeune Diplômé, est embauché pour participer à la création de la nouvelle application. Il avait déjà mis en place le serveur d'application utilisant la PLM Java avec Playframework. Le serveur hébergé à Telecom Nancy est fonctionnel. Il a mis en place la page d'accueil permettant de choisir le monde. Si le monde est implémenté, on avait accès aux instructions, à l'API, au player, à la zone d'écriture du code, à son exécution et à la démonstration. Il a permis l'exécution du code en intégrant une relation entre l'application web et le serveur d'application.

Il avait implémenté le monde des Buggles et tous les autres qui utilisent ce même monde pour affichage (termites, labyrinthe). Ma mission consiste à implémenter les mondes suivants afin qu'il travaille sur d'autres objectifs comme l'authentification et la récupération du travail des élèves.

2. Besoins et organisation

Afin d'accomplir ma mission, j'ai dû apprendre des nouvelles technologies et méthodes de travail. Il a fallu aussi comprendre les parties de l'architecture de l'application et les liens entre elles. Dès le premier jour, je me suis familiarisé avec des outils d'organisation et de travaux collaboratifs :

- **reporting :**

Martin Quinson m'a présenté le reporting, une méthode de travail et d'organisation. Elle consiste à écrire chaque semaine ou chaque mois ce que l'on a fait, les problèmes que l'on a rencontrés et les tâches planifiées. Personnellement, je le tenais à jour quotidiennement.

Cela permet de se rappeler là où on en est après un week-end par exemple. Grâce au type de fichier org et emacs, on peut structurer notre reporting en titres par importance. En effet, ils sont classés par nombre d'étoiles, plus il y en a moins le titre est important. Chaque titre peut ainsi se fermer ou se dérouler afin de faciliter la lecture. Il est aussi possible de placer des cases à cocher grâce à deux crochets, ainsi on peut cocher ce qu'on a fait dans les missions à accomplir. Sur Github, les fichiers de reporting s'affichent automatiquement en prenant en compte les titres.

Annexe 1 page 35 : Capture écran de mon reporting

- **GitHub :**

Afin que chacun puisse contribuer efficacement au projet, Git a été choisi. J'ai déjà eu la chance de l'utiliser par le passé mais lors du stage, je suis allé plus loin dans son utilisation. J'ai tout d'abord travaillé par branche. En effet, afin de ne pas nuire au projet fonctionnel, je créais une nouvelle branche, c'est-à-dire que je peux toujours accéder à la version actuelle et mes changements sont séparés. Si ce que j'ai réalisé est utilisable alors je pouvais pusher sur mon dépôt et proposer une pull request c'est-à-dire une demande de fusion avec le dépôt principal. Grâce aux branches, une par monde, chaque monde est ajouté indépendamment des autres. J'ai aussi utilisé le stash, permettant de manipuler facilement les modifications effectuées. De plus, l'outil meld associé avec la commande git difftool permet de voir rapidement les changements entre ce qui a déjà été commité et ce qui ne l'est pas encore. Ainsi je peux supprimer les espaces ou les lignes indésirables. A la place d'utiliser BitBucket, dont je me servais à l'IUT, j'ai utilisé GitHub. Il permet d'afficher le reporting au contraire de son concurrent et est intuitif.

Illustration 17: Logo de GitHub.

Ensuite, j'ai dû apprendre les technologies utilisées pour développer l'application :

- **angularJS :**

La nouvelle application web utilise donc le javascript afin de ne pas être obligée de recharger la page (HTML et CSS) à chaque interaction de l'utilisateur. Elle se sert plus précisément du framework angularJS. J'ai appris les bases du javascript à l'IUT et plusieurs frameworks mais pas celles de l'AngularJS. Le premier jour, j'ai suivi les cours et réalisé les exercices de Codeschool sur l'AngularJS. Je conseille ce site en anglais qui m'a permis de vite apprendre et de m'expérimenter. De plus, Matthieu m'a fourni la liste de règles qu'il a suivi afin d'avoir un code propre et ordonné.

Illustration 18: Logo d'AngularJS.

- **canvas :**

Le canvas est un composant HTML qui permet d'ajouter un contenu dynamique par des scripts qui dessinent ou insèrent directement des images. J'ai appris le canvas au fur et à mesure du stage en lisant des tutoriaux sur le net. Au cours des mondes que je devais dessiner, j'ai dû afficher différentes figures et j'allais rechercher les fonctions correspondantes dans l'API canvas.

Illustration 19: HTML 5 Canvas.

Ensuite, j'ai aussi confirmé des connaissances avec des langages qu'on m'a enseigné en cours :

- **Java :**

Le Java est le langage de programmation utilisé pour la PLM originale. C'est un langage orienté objet que j'ai étudié dès ma première année de DUT et il m'a servi lors de ce stage. En effet, j'ai dû ajouter des classes Java dans l'ancien PLM car celui-ci est encore opérationnel dans le service web.

Illustration 20: Logo Java.

- **Scala :**

Le Scala est un langage fonctionnel de programmation au contraire du Java, c'est-à-dire qu'il peut se composer uniquement de fonctions. Toutefois, la création d'objets et de classes reste possible. J'ai eu des cours de Scala pendant ma première partie de 2e année. Le framework Play utilise le Scala pour mettre en place le serveur d'application et faire le lien entre l'ancien et le nouveau PLM.

Illustration 21: Logo Scala.

Enfin, avant de commencer le développement, j'ai dû analyser et comprendre la conception choisie, notamment les relations entre le client et le serveur d'application :

- **playframework :**

Afin d'utiliser l'ancien PLM comme un serveur d'application, le framework Play a été choisi. Il permet de faire la relation avec le client web. Lors de l'exécution du code écrit par l'apprenti, ce code est envoyé au serveur play. L'ancien PLM va donc le compiler et l'exécuter. Pendant ce temps-là, le serveur va envoyer des opérations au fur et à mesure. Les opérations sont les méthodes principales permettant de réussir l'exercice. Lorsque la première PLM a fini son exécution, elle envoie un message de fin (résultat) que le client web pourra afficher.

Illustration 22: Logo Play.

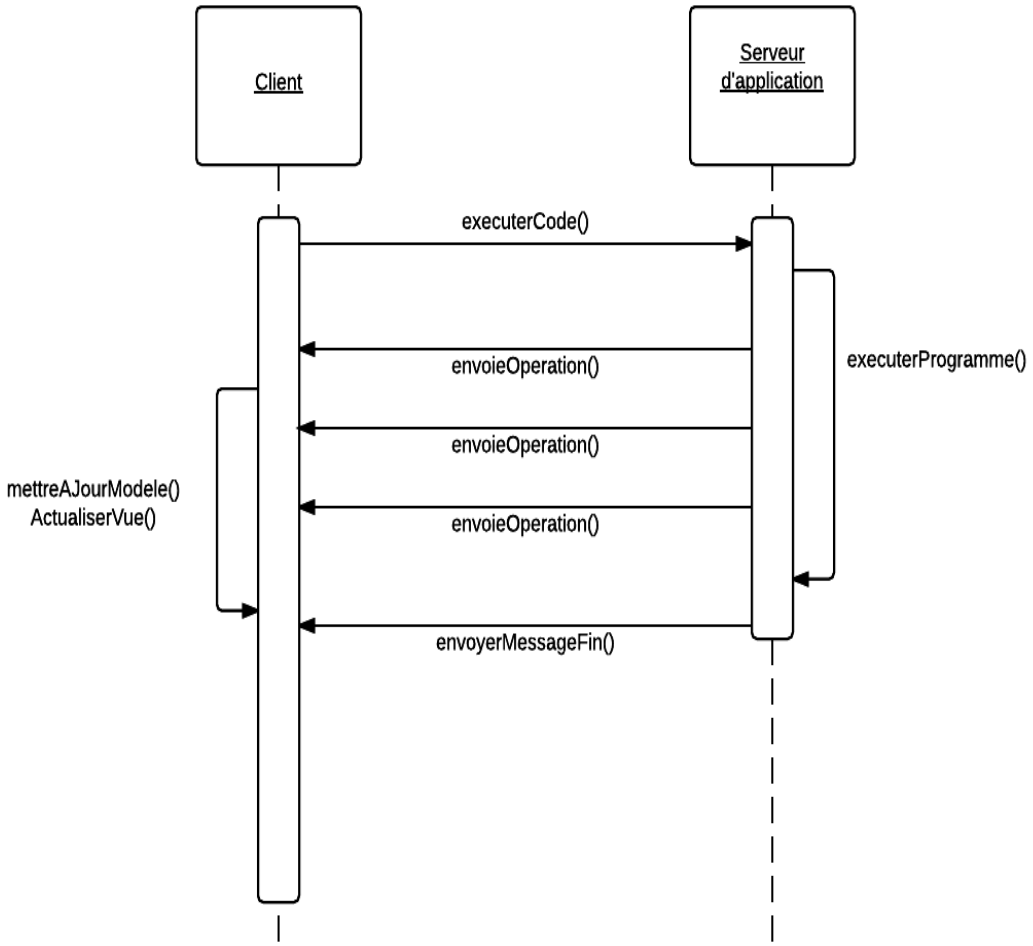


Illustration 23: Diagramme de séquence de l'excution du code de l'étudiant.

Enfin, afin de mieux visualiser la répartition du travail durant mon stage, je vous propose en annexe, un diagramme de Gantt :

Annexe 2 page 36 : Diagramme de Gantt

3. Conception et Réalisation

1. Pré-requis pour chaque monde :

L'application PLM en Java étant toujours utilisée comme serveur où sont stockées notamment les données des mondes, j'ai créé de nouvelles classes Java représentant les opérations pour chaque monde. Ces classes permettent de récuperer le nom et les attributs de l'opération, l'entité qui a réalisé celle-ci. En effet, ce sont ces classes opérations qui sont envoyées au client. Chaque opération va être traitée et le contrôleur peut modifier le modèle en conséquence et par la même occasion actualiser les vues. De plus, lorsque l'ancienne PLM va appeler une de ses méthodes correspondantes à une opération, j'ai dû lui ajouter une méthode permettant d'ajouter un objet opération (objet provenant des nouvelles classes que j'ai écrites). Enfin, l'appel de la méthode stepUI() permet de notifier le client qu'il y a eu des changements et il va faire appel aux services de dessin et donc actualiser les modifications d'affichage.

Afin de pouvoir utiliser les opérations, le monde et les entités, on se sert du convertisseur Scala du serveur Playframework, j'ai donc dû écrire des objets permettant de réaliser cette conversion en Json. En effet, même si la PLM originale conserve ses modèles, il est nécessaire de les écrire aussi sur la partie cliente en AngularJS sous forme de **factory***. Afin de faire le lien entre les deux membres de la relation, les factory vont utiliser les données en Json précédemment converties.

Certains modèles nécessitent plus de réflexion que d'autres car ils devaient convertir des collections non disponibles en Json. Pour un Vector de pancake par exemple, j'ai été obligé de créer une méthode de conversion implicite d'objet pancake puis traiter ce Vector comme un tableau grâce à Scala. Enfin, j'ai appliqué chaque élément du tableau à ma nouvelle méthode.

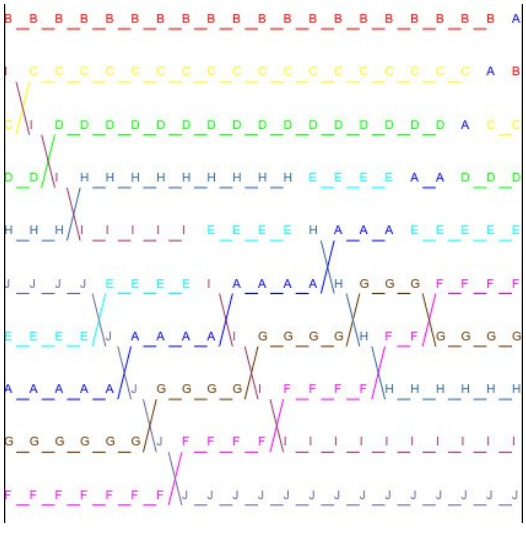
Afin de faciliter l'ajout de mondes, nous avons aussi modifié le contrôleur. En effet, la page web va appeler une fonction qui va parcourir un tableau dans le but de générer le bon monde avec la vue appropriée. Donc, pour chaque monde, j'ajoute un cas avec ce tableau avec les données respectives du nouveau monde. Pour finir, je devais implémenter dans chaque opération une fonction apply exécutée lorsqu'une opération est lancée, par exemple avec le player. En outre, l'implémentation d'une fonction reverse est nécessaire lorsqu'on recule avec ce même outil.

Les vues représentent graphiquement chaque monde et font appel à un service qui a une fonction de dessin. Je me suis inspiré de l'affichage des mondes de la première PLM mais je n'ai pas suivi les algorithmes de ses fonctions de dessins. En effet, les méthodes java Swing divergent des fonctions Canvas. J'ai disposé d'une liberté sur la méthode de dessin et la programmation des fonctions.

2. Le monde des tris :

Comme vu précédemment, le monde des tris contient un diagramme en barres à trier. Premièrement, il a fallu afficher le nombre de lectures et écritures. J'ai récupéré donc ces valeurs à partir du monde et je les ai écrites dans le coin supérieur gauche grâce à une méthode d'écriture de texte. Certaines opérations comptabilisent des lectures sans pour autant changer l'affichage. J'ai alors testé si je devais actualiser le nombre de lectures à chacune de ces opérations ou seulement toutes les prendre en compte lorsqu'une opération d'écriture arrivait. Modifier le dessin à chaque lecture s'est avéré trop long lors de l'exécution de toutes les opérations. J'ai donc choisi de n'actualiser la vue que lors d'une opération modifiant le tableau à trier (écriture). Pour afficher les rectangles, j'ai tout d'abord divisé la longueur du canvas par le nombre de valeurs du tableau. Puis, pour chaque valeur du tableau, je dessine son rectangle. Sa hauteur est représentée par la valeur traitée du tableau. Il existe en canvas une méthode pour dessiner les rectangles ayant comme paramètre les coordonnées du coin supérieur gauche, sa largeur et sa longueur. Si la place du rectangle est égale à sa valeur alors il est dessiné en vert sinon en rouge (mauvaise position). Pour changer de couleur, il existe une méthode prenant une couleur sous format RGB ou hexadécimal. De plus, si le nombre de valeurs du tableau n'est pas trop important, une lettre apparaît dans le celui-ci (de A à Z, dans l'ordre croissant). Le monde des tris possède aussi une deuxième vue permettant de visualiser chaque étape et chaque écriture dans une même fenêtre :

Illustration 24: Seconde vue du monde



Les étapes sont représentées en colonnes de gauche à droite. Chaque ligne illustre une valeur qui peut être inversée, on va donc voir deux lignes se croiser. Elles peuvent être aussi copiées, la ligne va donc relier une autre. Si une valeur a été modifiée avec la méthode setValeur(), elle apparaîtra avec en rouge avec un point d'exclamation. Si elle a été lue avec getValeur(), elle sera violette avec un point d'interrogation. L'image ci-dessus représente le dernier stade de la seconde vue car tous les états sont dessinés. Les étapes sont dessinées au fur et à mesure de gauche à droite. La hauteur du canvas est divisée par le nombre de valeurs afin d'avoir le bon nombre de lignes. La largeur du canvas est divisée par le nombre d'opérations afin d'avoir le nombre correct de colonnes. Comme dans la première vue, si le nombre de valeurs est trop important, je n'affiche plus les lettres pour laisser de la place à plus de lignes. Afin de conserver chaque étape du tableau, j'ai déclaré un tableau qui les stocke à chaque opération apply et supprime à chaque reverse. Enfin, j'ai parcouru le tableau d'opérations pour traiter les changements. Cette vue m'a posé des problèmes de précision d'affichage et parcours de tableau (opérations, étapes). J'ai donc utilisé plusieurs indices afin que chaque ligne et colonne soit indépendante entre elles. Pour l'affichage, certains croisements de lignes, points d'interrogations et d'exclamations apparaissaient avant leur étape respective. J'ai ajouté un rectangle blanc de nettoyage qui se rétracte au fur et à mesure des étapes.

3. Le monde DutchFlag (drapeau hollandais) :

Le monde DutchFlag contient un tableau de valeur allant de 0 à 2, correspondant respectivement au bleu, au blanc et au rouge. J'ai donc divisé la hauteur du canvas par le nombre de valeurs, c'est-à-dire le nombre de lignes. La largeur des bandes reste celle du canvas. L'opération inverse donc deux lignes au fur et à mesure jusqu'à réussir à assembler le drapeau hollandais. J'ai aussi écrit dans en haut à droite le nombre de mouvements éffectués pour réussir l'exercice. Cette vue n'a pas été difficile car j'ai utilisé mon travail du monde des tris car elle a des points communs avec la vue classique de celui-ci.

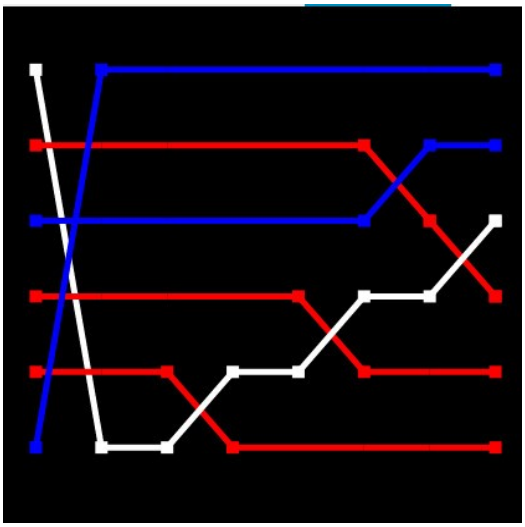


Illustration 25: Seconde vue du drapeau hollandais.

Par la suite, j'ai décidé d'ajouter une seconde vue au monde DutchFlag non présente dans la première version de PLM afin de pouvoir visualiser tous les changements dans un même affichage. Tout d'abord, j'ai colorié le fond en noir afin de voir les lignes blanches. J'utilise un tableau mémoire me permettant de stocker toutes les étapes. A chaque début, fin et croisement de lignes, j'ai ajouté un carré au bord de la ligne afin de mieux voir les changements. Les étapes s'affichent au fur et à mesure de gauche à droite comme la seconde vue du monde des tris dont je me suis inspiré.

4. Le monde Pancake :

Pour le monde Pancake, j'ai affiché tout d'abord le nombre de retournements réalisés. Puis j'ai dessiné des rectangles jaunes pour les pancakes. Pour cela, je divise la hauteur du canvas par le nombre de pancakes pour savoir leur hauteur. Pour la largeur, je récupère la valeur de chaque pancake. Si je suis dans un exercice avec des brulés, j'ajoute un rectangle brun en fonction d'un booléen qui indique s'il est sur la face du dessus ou du dessous. Pour dessiner la spatule qui révèle la dernière pile de pancakes retournée, j'ai tracé des lignes. Je choisis un point avec ses coordonnées avec moveTo(x,y) et un autre avec lineTo(x,y) puis j'utilise stroke() pour tracer. Avec lineWidth je peux choisir l'épaisseur de la ligne. Enfin, pour chaque pancake, je vérifie si son voisin du dessus a une valeur d'une unité supérieure.

Si c'est le cas, je dessine un petit rond rose entre ces deux pancakes grâce à un arc qui prend en paramètre les coordonnées du centre, le rayon, l'angle de départ (conseillé à 0) et l'angle de fin (2 * PI pour dessiner un cercle complet). Par contre pour remplir à la place de tracer, on utilise fill() à la place de stroke().

5. Le monde Baseball :

C'est celui qui m'a apporté le plus de difficultés mais je suis satisfait car j'ai utilisé des formules de géométrie. Ce monde contient un tableau de valeurs représentant les couleurs des Buggles. Tout d'abord, j'ai écrit en haut à droite le nombre de mouvements exécutés. Puis j'ai dessiné la pelouse, en effet, j'ai dessiné un grand carré de la taille du canvas vert. Ensuite, j'ai tracé un rond brun au centre afin de réaliser le terrain de jeu.

Afin de placer les bases près du périmètre du cercle à égale distance l'une de l'autre, j'ai décidé de fonctionner avec un système de points sur un cercle de même centre mais de rayon légèrement inférieur. Pour dessiner les positions à l'intérieur des cercles, j'ai trouvé le segment entre les milieux des bords des bases et j'ai placé leurs centres sur celui-ci. Chaque buggle a une couleur et s'il correspond avec la couleur de la base où il est placé, je trace un cercle gris autour sinon rouge. Pour dessiner les flèches indiquant les déplacements, j'ai utilisé la méthode quadraticCurveTo. Elle prend en paramètre les coordonnées du point de contrôle Bézier et celles du dernier point de la courbe. Il ne faut pas oublier de se placer au premier point de cette courbe avec un moveTo. Le point de la nouvelle position du Buggle n'a pas été évidente à trouver à chaque fois. En effet, si je passais d'opérations en opérations avec le player tranquillement, il n'y avait pas de soucis. Mais je jouais avec le player de manière abusive (aller-retour successif ou clique sur sa barre), il arrivait que le bon point ne soit pas bien trouvé par mon algorithme. J'ai donc décidé de placer tous les points des buggles dans un tableau et rechercher dans celui-ci le point correspondant. Cela a corrigé le problème. Pour la tête de la flèche, j'ai rejoint trois points en respectant le même angle afin de faire un triangle isocèle.

Le monde Baseball a aussi une seconde vue pour voir tous les mouvements des Buggles dans une seule fenêtre. J'ai tout d'abord dessiné les bases à gauche et à droite du canvas. J'ai fait un tableau gardant les différentes étapes. Par contre, les bases sont toujours reliées et sont modifiées au fur et à mesure.

6. Le monde des tours d'Hanoï :

Tout d'abord, j'ai dessiné trois lignes représentant les tours d'Hanoï à distances égales. Les pièces à bouger sont des valeurs entières dans trois tableaux. A chaque opération, je les parcours et je dessine une pièce par valeurs. Les hauteurs sont toujours identiques mais les largeurs sont proportionnelles aux valeurs. En haut, à gauche, j'ai inscrit le nombre de mouvements effectués.

7. Les opérations :

Pour chaque monde, il faut implémenter les vues mais aussi les opérations et plus précisément les méthodes apply et reverse. L'apply est l'exécution d'une opération et le reverse est son contraire afin de l'annuler.

Ce système est crucial pour la mise en place du player qui permet d'avancer ou de reculer dans l'exécution des opérations. Chaque opération a été ajoutée dans le serveur d'application PLM grâce à des classes Java comme vu dans la partie pré-requis. Les attributs de cette opération sont convertis en format Json et ces informations sont utilisées dans les factory d'opérations. Par exemple, pour une opération de copie, on trouve généralement la source et le destinataire. L'apply va donc modifier le tableau du monde en remplaçant la valeur du tableau à l'index destination par la valeur à l'index source. Lors d'un échange de valeurs, il est nécessaire d'utiliser une variable intermédiaire qui va stocker la valeur écrasée afin de l'ajouter à l'autre index. Il est possible d'ajouter des attributs dans la factory par exemple pour garder des valeurs en mémoire. Cela permet de réaliser le reverse en modifiant le tableau avec les valeurs sauvegardées. Pour les secondes vues, je dois conserver les étapes effectuées donc j'ajoute la nouvelle étape au tableau du monde grâce à l'apply. Avec la reverse, je supprime le dernier élément de ce tableau. Pour le nombre de mouvements, d'écritures ou de lectures, j'incrémente un attribut bien précis dans un apply et le décrémente pour un reverse. Il est aussi possible d'employer un booléen qu'on change à faux ou à vrai afin de choisir si on veut afficher des éléments seulement en apply par exemple.

8. Futurs portages :

Pendant la dernière partie de mon stage, j'ai travaillé sur le portage du monde des tortues ainsi que celui du Logo puisqu'il partage le même monde (monde, opérations, vue). Dans la première PLM, le choix des exercices du monde des tortues s'effectuait grâce à un système d'images cliquables. En effet, elles représentaient une figure à dessiner. Dans la web PLM, cette méthode de choix n'est pas opérationnelle et il serait plus simple de garder le choix classique des exercices. De plus, implémenter un mode de dessin libre est une bonne idée afin de laisser l'utilisateur s'entraîner à sa convenance sans le bloquer. J'ai ajouté une classe d'opération move dans la PLM originale et les objets Scala nécessaires dans la version web. J'ai commencé à implémenter l'apply et le reverse de l'opération ainsi que la vue mais ceci n'est pas complet. En effet, la tortue ne dessine pas et ne se tourne pas (pas de rotation). En outre, les mondes du Lander et du Lightbot restent à porter. Afin d'éviter toute mauvaise surprise, il est crucial de réaliser une série de tests unitaires sur mon travail car je n'ai réalisé que des essais utilisateurs.

IV. Conclusion

Ce stage m'a permis de découvrir le monde professionnel mais aussi celui de la recherche informatique. J'ai eu la chance d'assister à une soutenance de thèse ainsi que des séminaires Ingénieurs Jeune Diplômé. J'ai aussi testé un jeu sur les transferts d'argent dont le but est d'étudier la confiance entre collaborateurs.

Au cours de ma mission, j'ai eu la chance d'apprendre les bases d'AngularJS, d'acquérir une bonne maîtrise du canvas et d'utiliser mes connaissances en Java et en Scala. Participer au développement d'un projet collaboratif a amélioré mon utilisation de Git. De plus, j'ai acquis une meilleure méthode de travail avec la tenue d'un reporting. Je retiendrai que lors de la réflexion d'un algorithme, il ne faut pas prendre seulement en compte les problèmes actuels mais ceux qui pourront survenir.

Je suis très fier d'avoir participé au développement de la web PLM car c'est un projet d'une immense richesse apportant de la connaissance. Je conseille de le proposer aux élèves de Terminale venant à la porte ouverte de l'IUT Charlemagne. De cette manière, ils pourront voir si la programmation leur plaît et commencer dès à présent leur apprentissage. Enfin, je l'utiliserai afin de réviser mes acquis en programmation après une période de vacances.

Ayant apprécié l'ambiance générale et le fort potentiel pédagogique du LORIA, je n'hésiterai pas à postuler ici pour de prochains stages.

Bibliographie

Document :

Programmer's Learning Machine Campagne ADT 2014, 13 pages

Sites :

LORIA : 3 juin 2015, www.loria.fr/

LucidChart : du 8 au 12 juin 2015, https://www.lucidchart.com/

Mozilla Developer Network : du 8 au 12 juin, https://developer.mozilla.org/fr/docs/

Glossaire

Travis CI : Service réalisant une série de tests sur les projets hébergés sur GitHub

Debugger : Permet d'exécuter un programme pas à pas, d'afficher les valeurs des variables et de mettre des points d'arrêts afin d'identifier d'éventuelles erreurs.

MVC : L'architecture MVC se compose de trois parties interconnectées : le modèle, la vue et le contrôleur afin de mettre en place une application interactive.

WebSocket : Protocole réseau permettant la liaison entre client et serveur.

Logs : Journal permettant de stocker un historique des évènements.

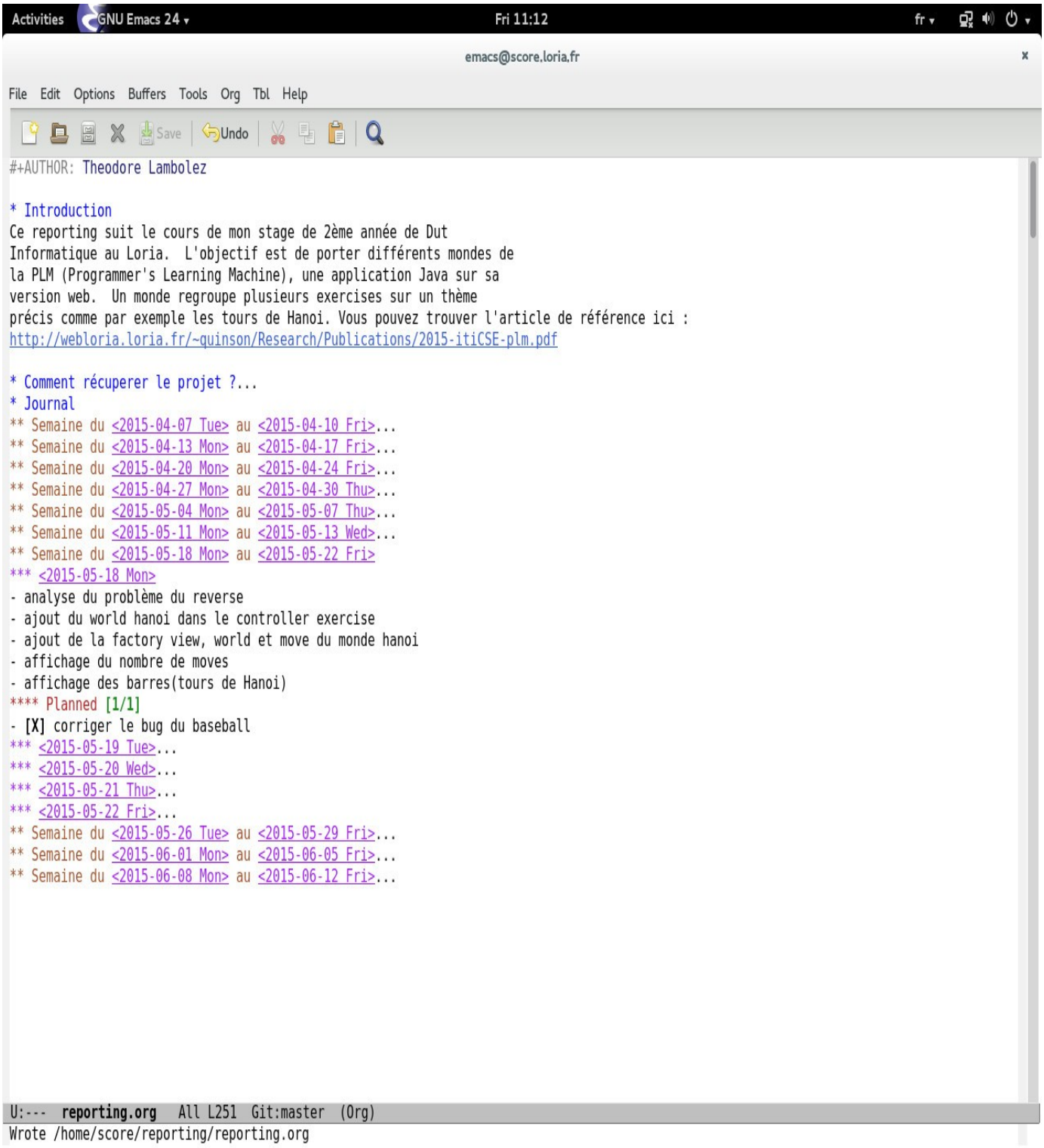
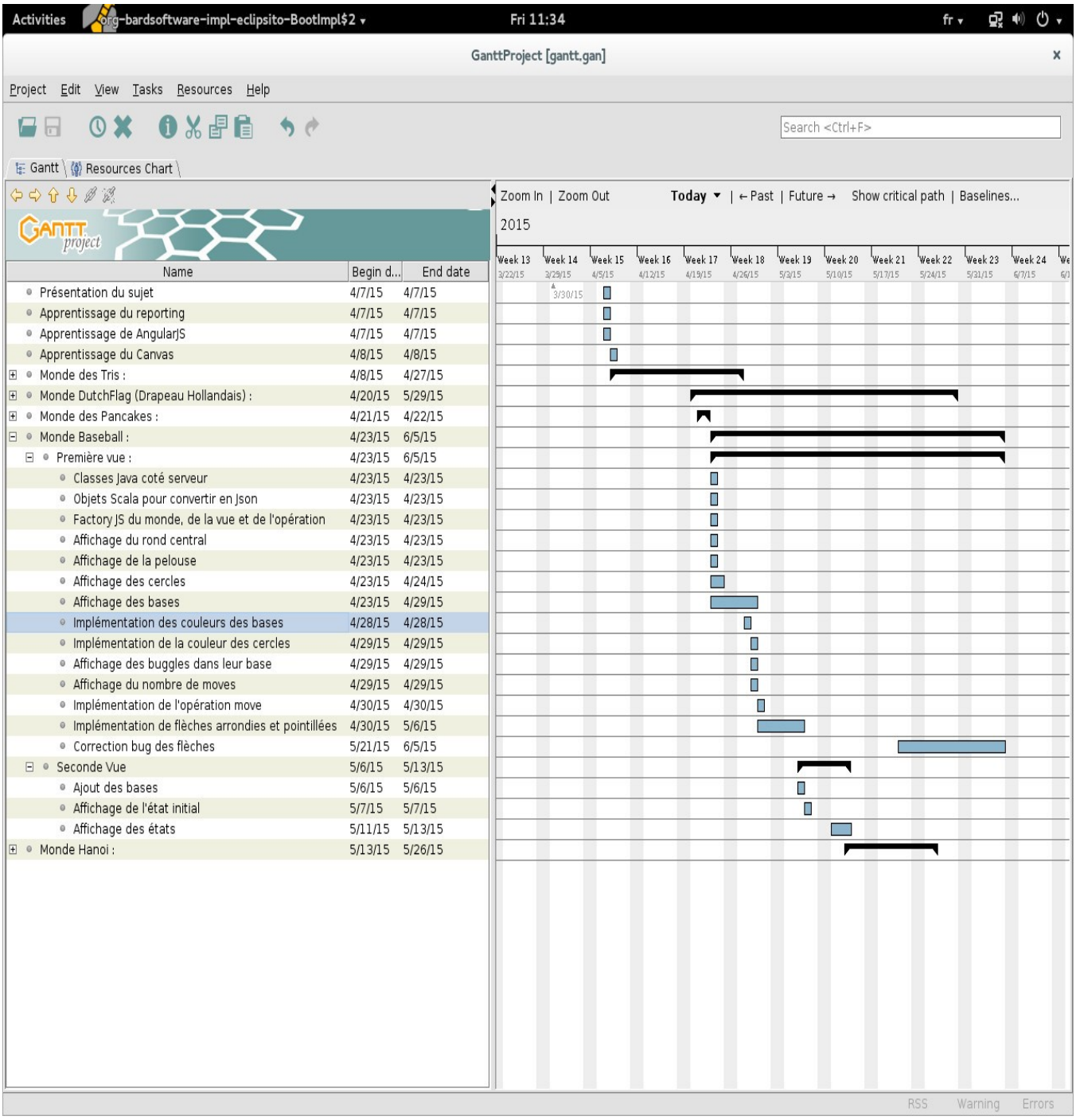
Factory : Service AngularJS c'est-à-dire objet pouvant être injecté comme dépendance, se comportant comme une fonction Javascript habituelle.

Table des annexes

Annexe 1 : Capture d'écran de mon reporting…….35

Annexe 2 : Diagramme de Gantt…….36

Annexe 1 : Capture d'écran de mon reporting



Annexe 2 : Diagramme de Gantt