Dossier d'analyse et conception

I Hear, I Say

Projet Tutoré 2016 - Groupe M

Réalisé par :

Guillaume Robert

Yoann Gathignol

Titouan Bouete-Giraud

Destinataire :

Isabelle Clavel

Superviseur :

Laurence Redon

BLAGNAC

iut Midi-Pyrénées

UNIVERSITÉ TOULOUSE Jean Jaurès

Table des matières

Introduction

Ce projet tutoré est une demande d'Isabelle Clavel, professeure d'anglais au département Informatique de l'Institut Universitaire Technologique de Blagnac. Laurence Redon, également enseignante à l'Institut, supervise ce projet.

Dans le cadre de son enseignement, Mme Clavel utilise des grilles d'exercices de prononciation qu'elle crée elle-même à l'aide de logiciel de tableur tel que Microsoft Office Excel. Bien que très complet, ce type de logiciel n'a pas pour but premier ce type d'exercice, et peut donc présenter diverses contraintes pour l'utilisateur :

- De temps : l'absence de format standard force l'utilisateur à dédier une partie non négligeable de son temps à gérer la forme de la grille.
- De suivis : La modification de grille peut s'avérer très fastidieuse à cause des contraintes sur les données saisies, à tel point qu'il peut s'avérer préférable de commencer une nouvelle grille.

Le client a donc demandé une solution alternative qui lui permettrait un gain de temps non seulement à la réalisation, mais aussi lors de la modification éventuelle de ces grilles. Pour de plus amples informations, nous invitons le lecteur à consulter le Cahier Des Charges Utilisateurs (CDCU) fournis avec ce dossier.

I hear	I say Start here	I hear	I say	I hear	I say	I hear	I say	I hear	I say	I hear	I say	I hear	I say
Ajax	HTML	Python	C++	HTML	Python	C++	Microsoft	Microsoft	PHP	PHP	•Net	•Net	MySQL
highlight	DBMS	Browse r	XML	Visual Basic	delete	MySQL	JQuery library	XML	Visual Basic	DBMS	Browse r	JQuery library	highlight
delete	Android	Yosemite	gigabyt e	Java	DCM	Apache	RSS flow	Android	Apache	gigabyt e	Java	RSS flow	Yosemite
Perl	Ruby	Adobe	SQL	Ruby	Adobe	SQL	API	Oracle	Perl	API	Ajax	DCM	Oracle

Figure 1 : Exemple de résultat

Sur cette figure on peut voir ce qui est attendu au final. Ce que nous devons permettre au client de générer grâce à notre application. En effet l'activité consiste à prononcer des mots de la colonne « I Say », celui qui a ce mot dans la colonne « I Hear » devra dire le mot correspondant et ainsi de suite. Cela forme une chaine jusqu'à ce que tous les mots aient été prononcés.

Comment faciliter la création de cette activité grâce à un logiciel dédié à cette tâche ?

I. Organisation actuelle et future

Actuellement, pour réaliser son activité, notre cliente doit passer par plusieurs étapes plus ou moins longues. Tout d'abord elle doit faire de la mise en forme afin de créer un tableau contenant deux colonnes : « **I Hear** » et « **I Say** ». Ensuite, elle doit entrer les mots un par un, en double puisque chaque mot doit apparaître et dans la colonne I Hear et dans la colonne I Say. Cependant comme le but de l'exercice est de dire tous les mots présents dans la grille, les mots ne sont pas entrés dans l'ordre mais dans un ordre aléatoire. C'est pourquoi la création est prend beaucoup de temps et la modification est aussi très longue. Par exemple, pour générer cette petite capture d'écran il aura fallu presque 5 minutes juste pour remplir correctement les grilles.

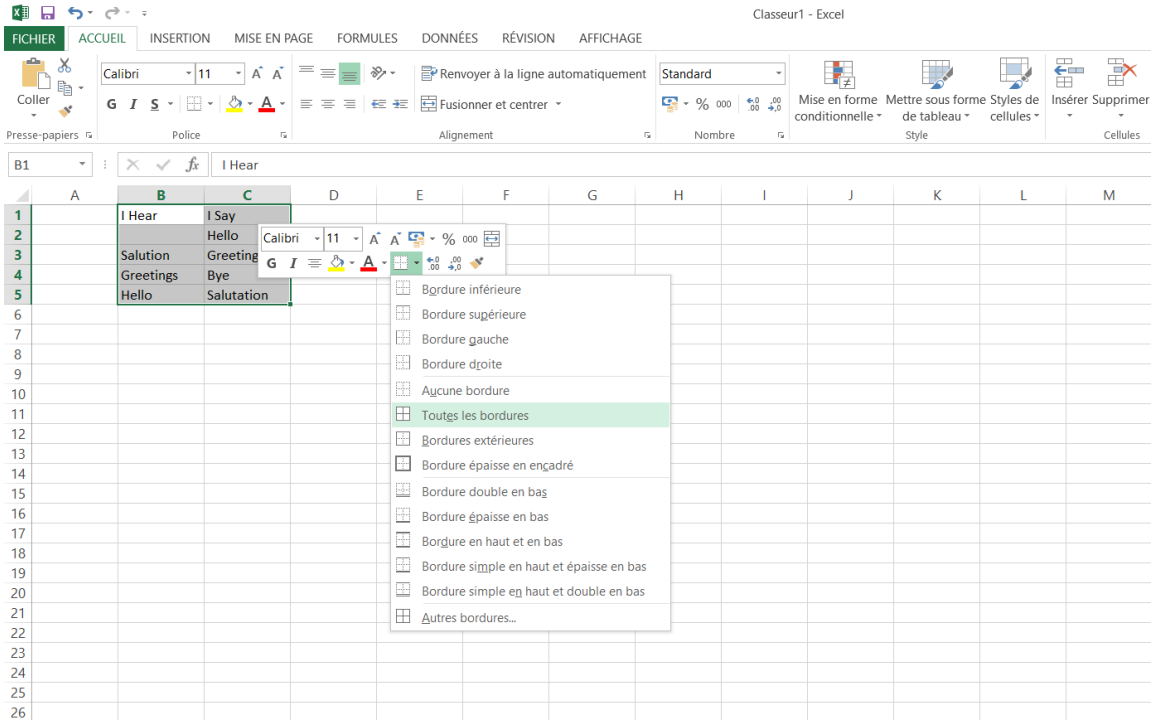


Figure 2 : Exemple de génération de grille

Grâce à l'application que nous produisons, il y aura une automatisation de plusieurs étapes lors de la création des grilles.

La modification d'une case entraînera automatiquement la modification de celle qui lui est associée (si on modifie une case de la colonne « I Say » contenant le mot « Now » alors la case contenant ce mot dans la colonne « I Hear » sera modifié).

Lors de la création d'une grille, une option permettra de créer la grille automatiquement sans avoir à faire la mise en forme et en aillant juste à entrer les mots les uns à la suite des autres.

Les grilles sous format .csv seront générées directement grâce à l'application, c'est-à-dire que la mise en forme sera faite automatiquement.

Ces choix ont été faits dans le but de faciliter la production de cette activité par notre cliente.

II. Analyse

Diagramme de cas d'utilisation

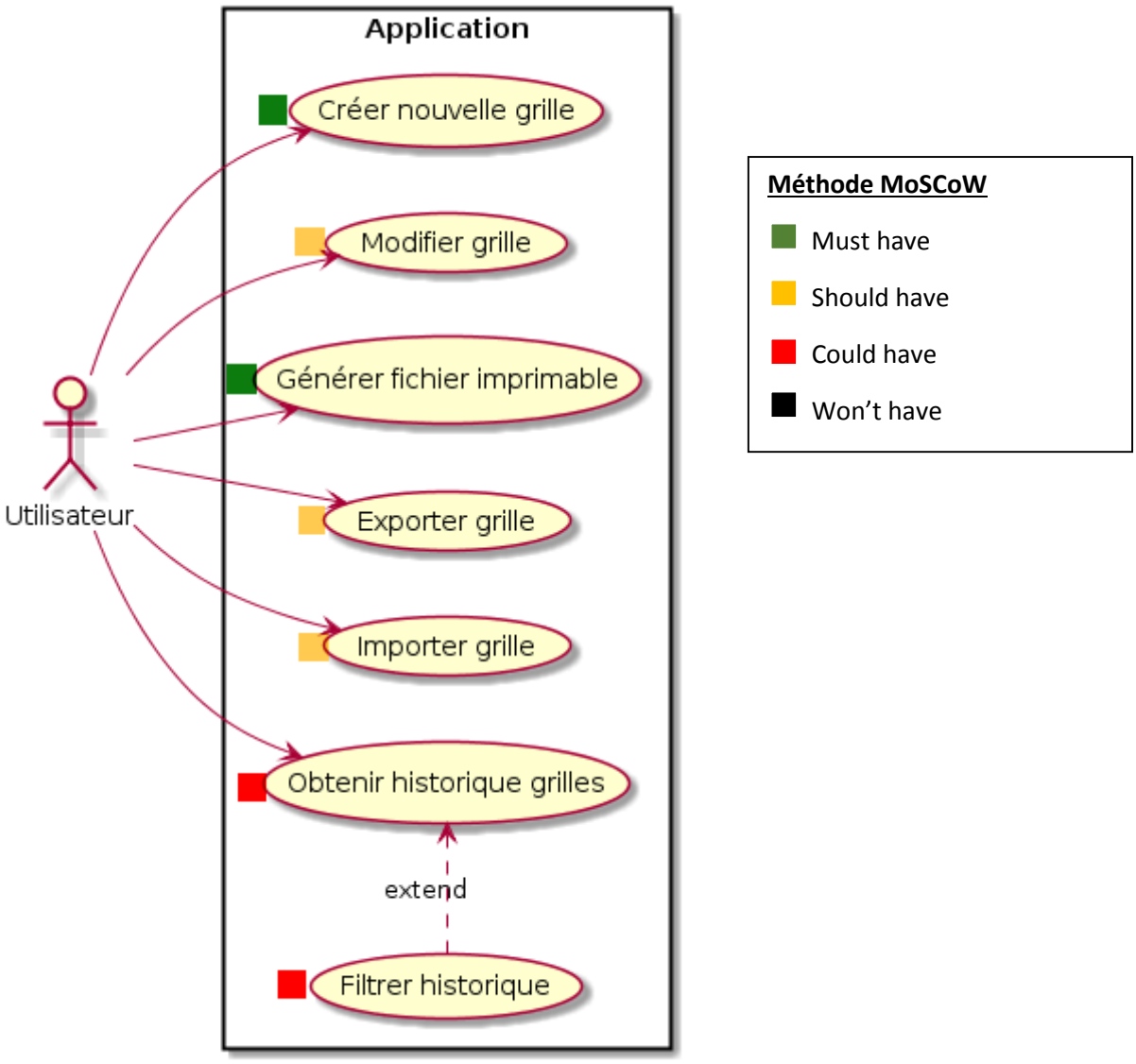


Figure 3 : Diagramme de cas d'utilisation

Ce diagramme des Cas d'Utilisation a été créé après l'analyse des besoins du client qui a ensuite validé ce diagramme. Pour l'organisation de la phase de développement, nous avons utilisé la méthode « MoSCoW » qui sert à hiérarchiser les besoin pour avoir le plus rapidement possible un prototype fonctionnel et pour un déroulement général efficace du projet.

En vert, nous avons donc les fonctionnalités primaires de l'application. La fonctionnalité « Créer nouvelle grille » est celle qui constitue la base pour l'implémentions de toutes les autres. La fonctionnalité « Générer fichier imprimable » est ensuite la fonctionnalité jugée la plus importante derrière la création de grilles.

En orange, nous avons les fonctionnalités nécessaires à l'application, celles-ci sont cependant dépendantes de la fonctionnalité « Créer nouvelle grille » et ne pourrons être implémentées seulement après celle-ci.

En rouge, nous avons les fonctionnalités qui ont été demandées par le client mais jugée moins urgente que les autres, et seront implémentées seulement après l'implémentation de toutes les autres fonctionnalités.

Il a été convenu avec le client et le superviseur que d'autres fonctionnalités pourraient être ajoutées si toutes celles-ci ont été implémentées avant la fin du projet.

Analyse du comportement de l'application

Dans cette partie nous allons voir les scénarios principaux des interactions entre l'utilisateur et le système.

a. Créer une grille

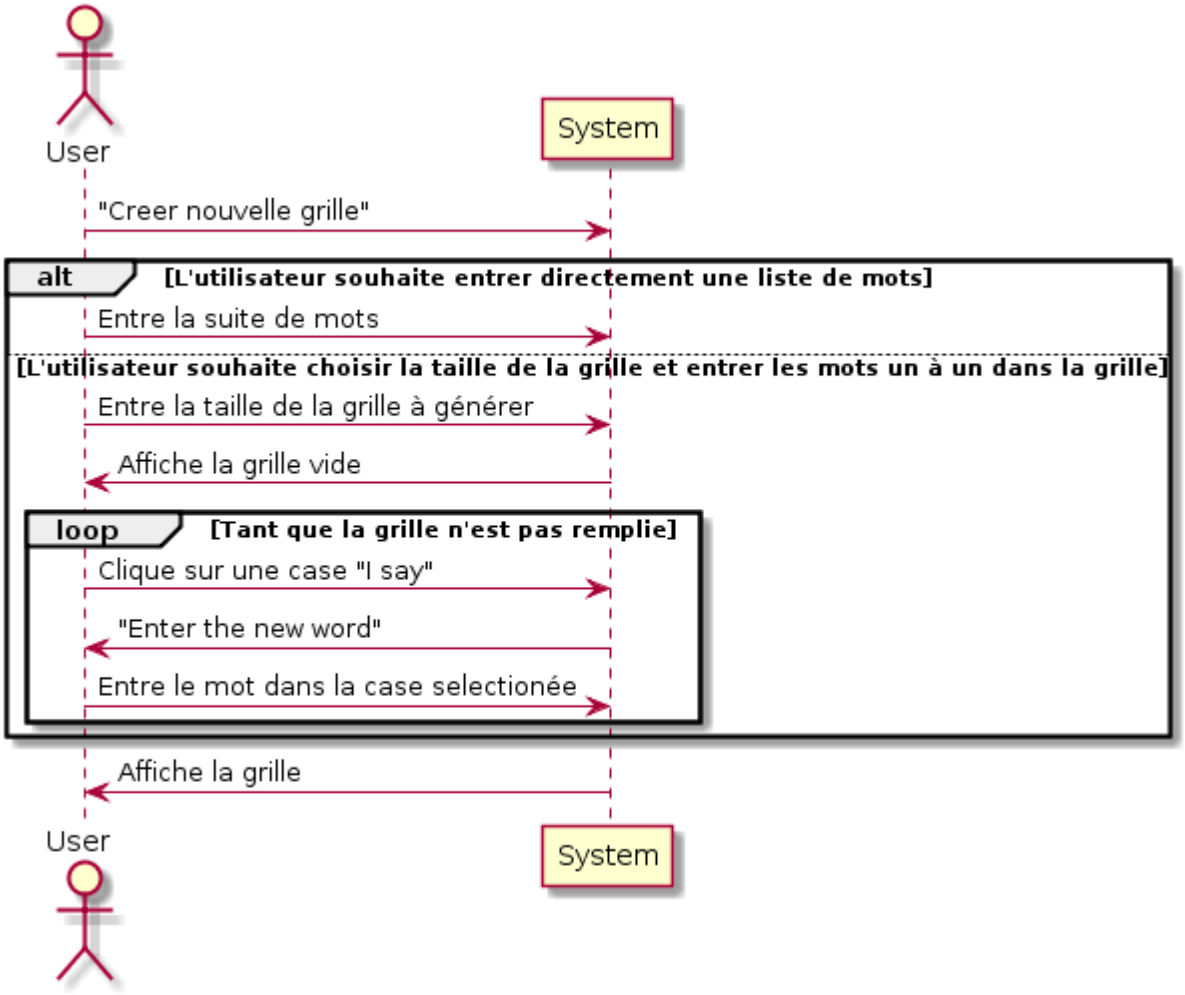


Figure 4 : Diagramme de séquence système Créer une grille

Ceci est le diagramme de séquence système dans le cas où l'utilisateur souhaiterait créer une tout nouvelle grille.

L'utilisateur a deux possibilités (exigés par le client) :

- la première étant de simplement rentrer en une seule fois toute la liste des mots qu'ils souhaitent retrouver dans la grille, et cette même grille se génèrera et se mélangera automatiquement. L'ordre de la chaine est aléatoire.

- la deuxième étant de choisir initialement une taille (nombre de lignes et nombre de colonnes) et de rentrer les mots un à un dans chaque case « I say ». La chaine se créera toute seule, et encore une fois, avec un ordre aléatoire.

b. Modifier une grille existante

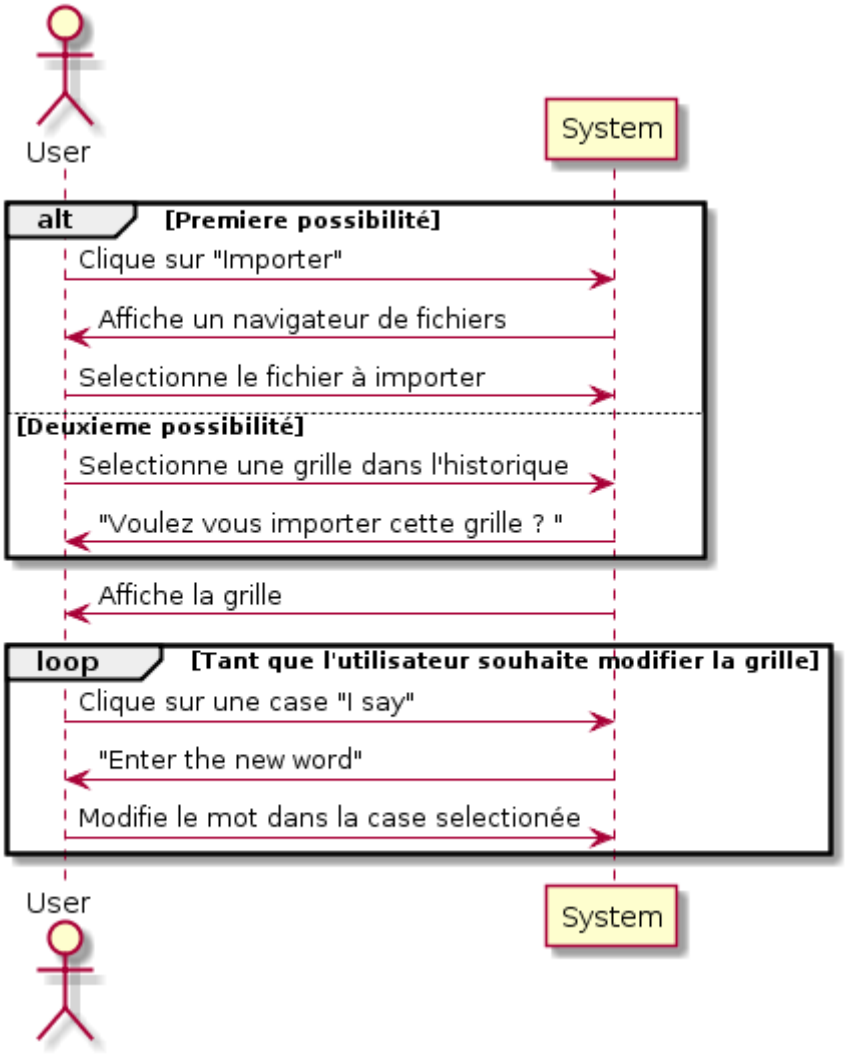


Figure 5 : Diagramme de séquence système Modifier une grille existante

Ceci est le diagramme de séquence système dans le cas où l'utilisateur souhaiterait modifier une grille existante.

L'utilisateur a deux possibilités :

- Cliquer sur le bouton importer qui ouvrira un navigateur de fichier, l'utilisateur pourra donc aller chercher le fichier à importer.

- Cliquer sur le nom de la grille dans l'historique de grilles qui importera cette dernière.

L'utilisateur n'aura donc qu'à sélectionner chaque case « I say » à modifier, et les cases « I hear » correspondantes se modifieront en conséquent.

Diagramme classes métier

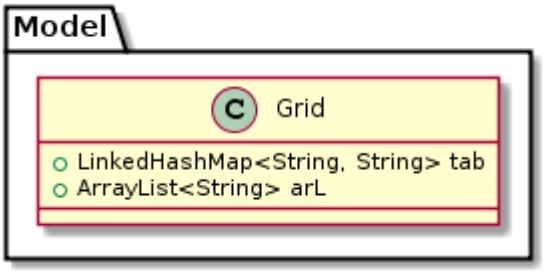


Figure 6 : Diagramme de classe métier

Ceci est le diagramme des classes métier.

Cette classe sert à stocker les chaines de caractères entrées dans l'interface pour créer le fichier sauvegardé.

L'ArrayList arL est une ArrayList de toutes les chaines entrées par l'utilisateur (celles des cases « I say »). C'est sur celle-ci que l'on va appliquer la méthode shuffle() pour mélanger les mots.

La LinkedHashMap tab nous sert à stocker les duos de mots (les doublons « I hear/I say »). Ce type de structure nous permet d'avoir une chaine de duo de mots, crée avec les chaines contenue dans arL (mélangée au préalable).

A noter, les structures de données choisies pourront être sujets à des modifications au fur et à mesure du projet si cela est nécessaire.

Maquette de l'interface

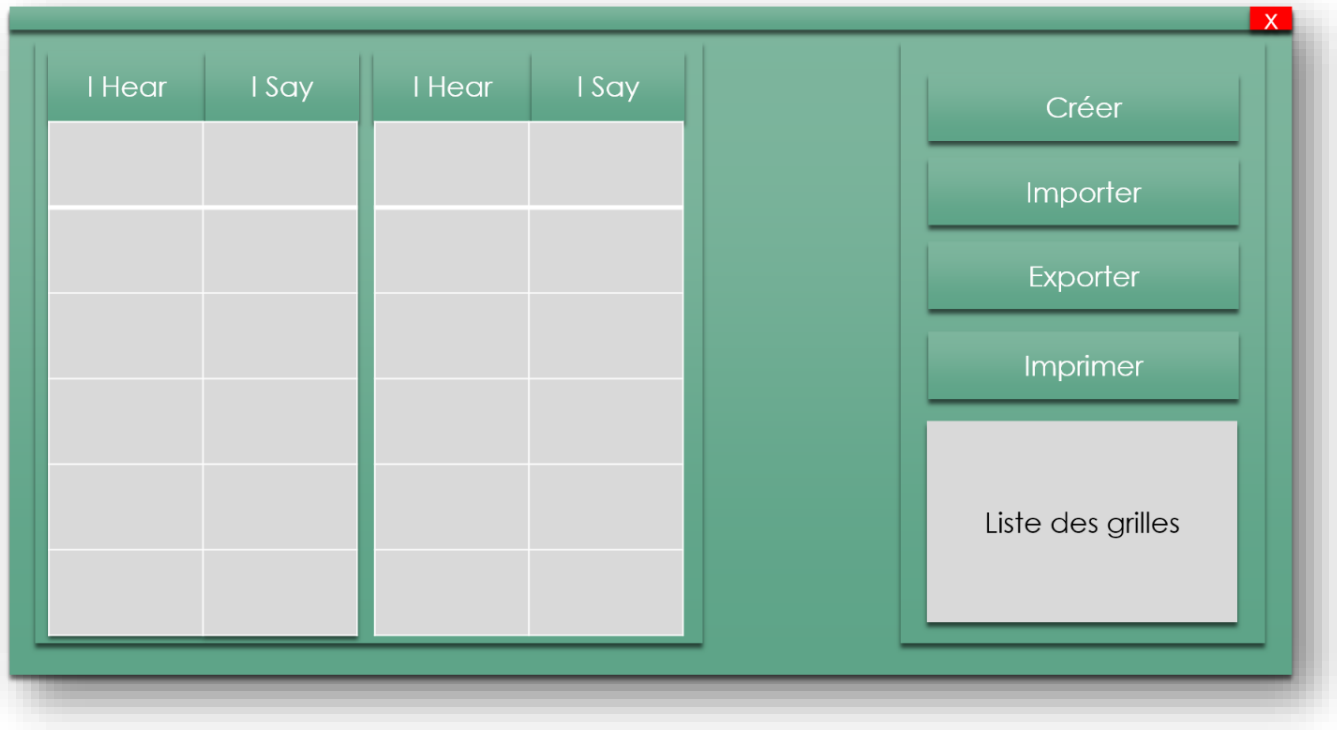


Figure 7 : Maquette

Interface actuelle

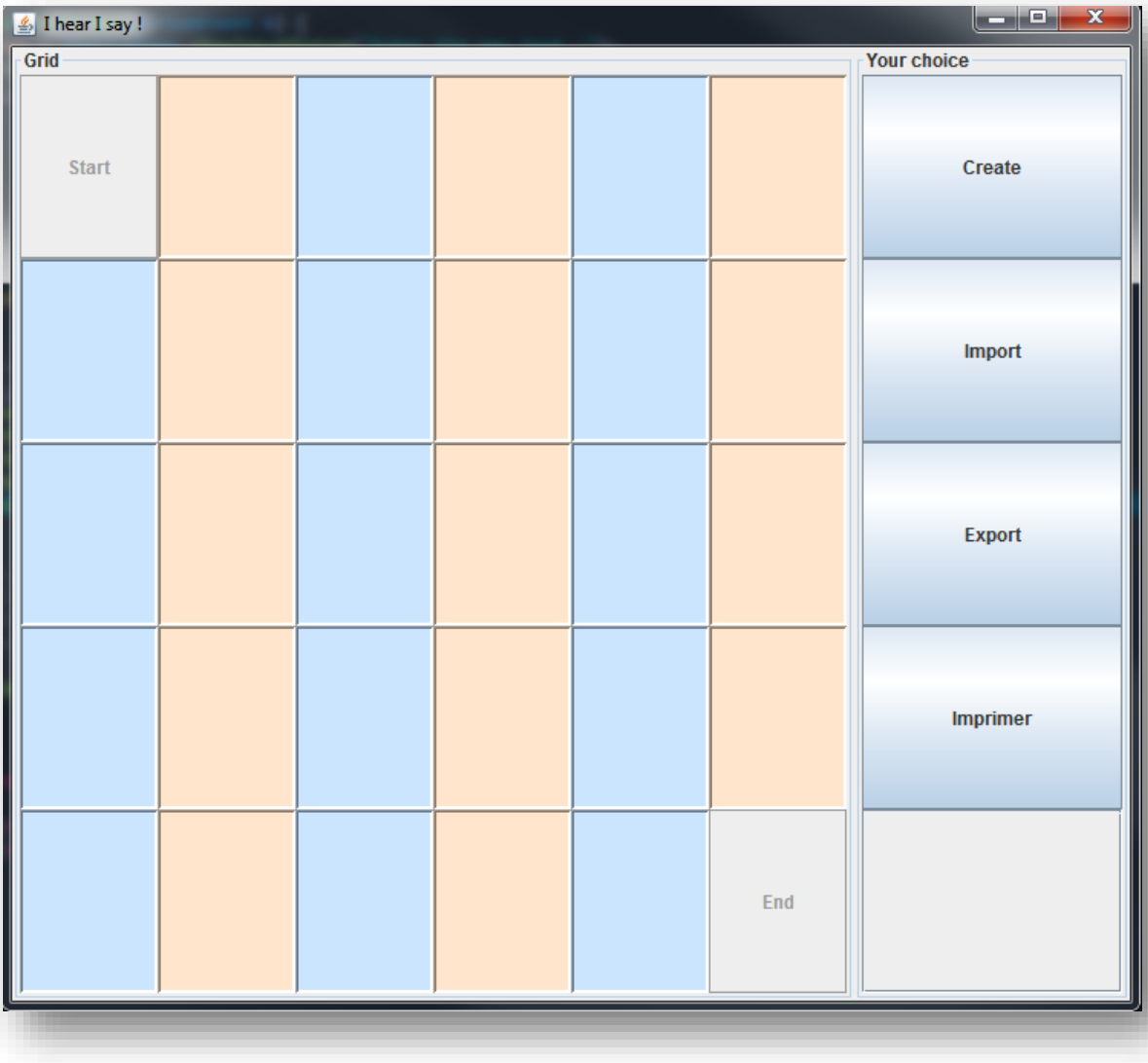


Figure 8 : Interface actuelle

Ci-dessus se trouve notre première maquette concernant l'application. La deuxième figure représente notre avancement en termes de l'interface. Les couleurs et le design ne sont pas vraiment représentatifs pour le moment puisque nous nous concentrons dans un premier temps sur les fonctionnalités importantes. Etant donné que notre client n'a pas défini le design comme un besoin vital, celui-ci sera implémenté après les fonctionnalités les plus importantes.

On retrouve donc les principaux boutons qui nous permettrons d'implémenter les fonctionnalités associées à ces-derniers. Sur la deuxième figure la colonne beige représentera la colonne « I Say » et la bleu « I Hear ». Il suffira de cliquer sur une case beige pour y modifier le mot. On peut le voir grâce à la figure suivante :

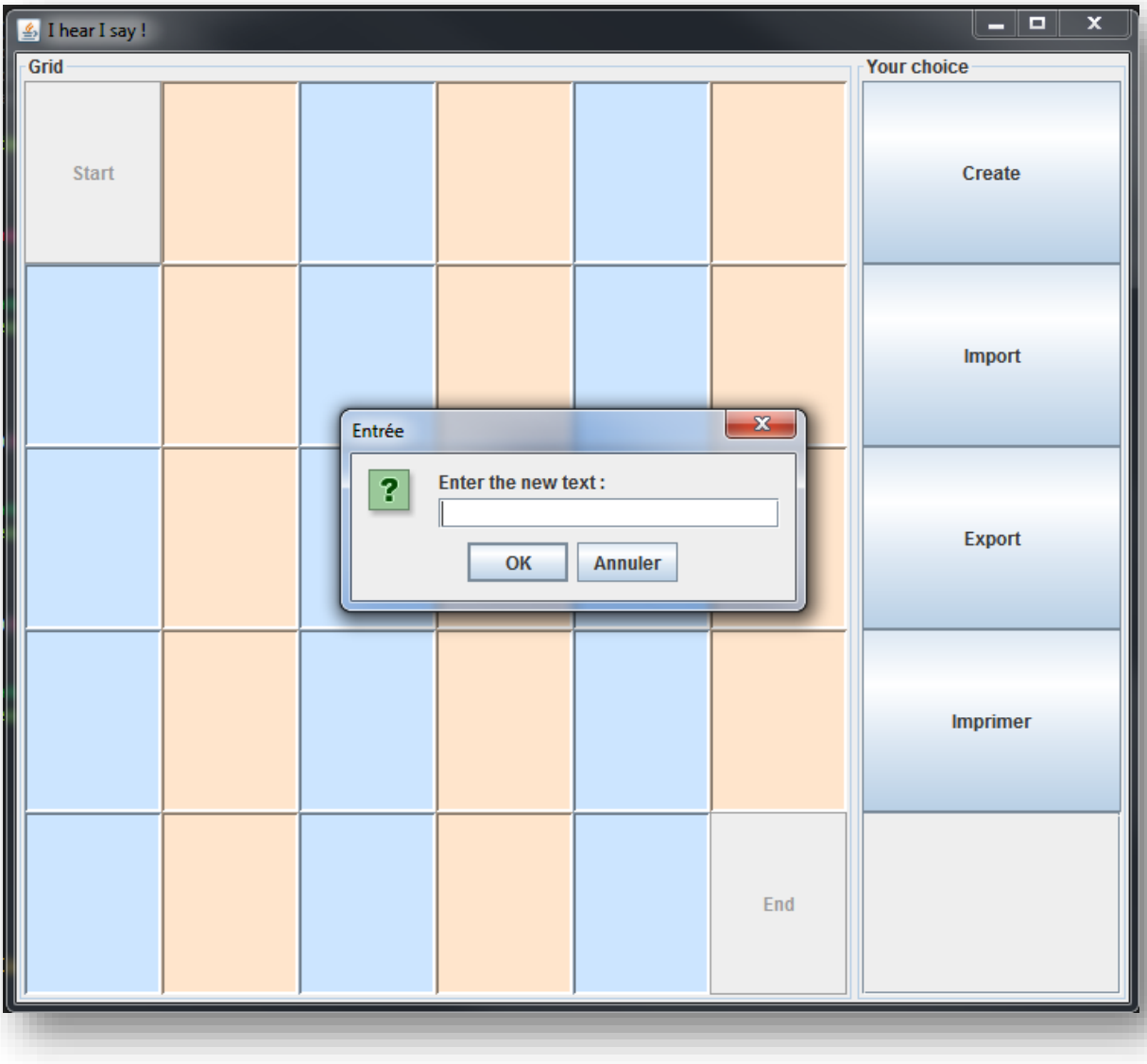


Figure 9 : Interface lors du clic sur une case

III. Conception

Diagramme de séquence et Diagramme de classes participantes pour le cas d'utilisation Créer nouvelle grille

a. Diagramme de séquence

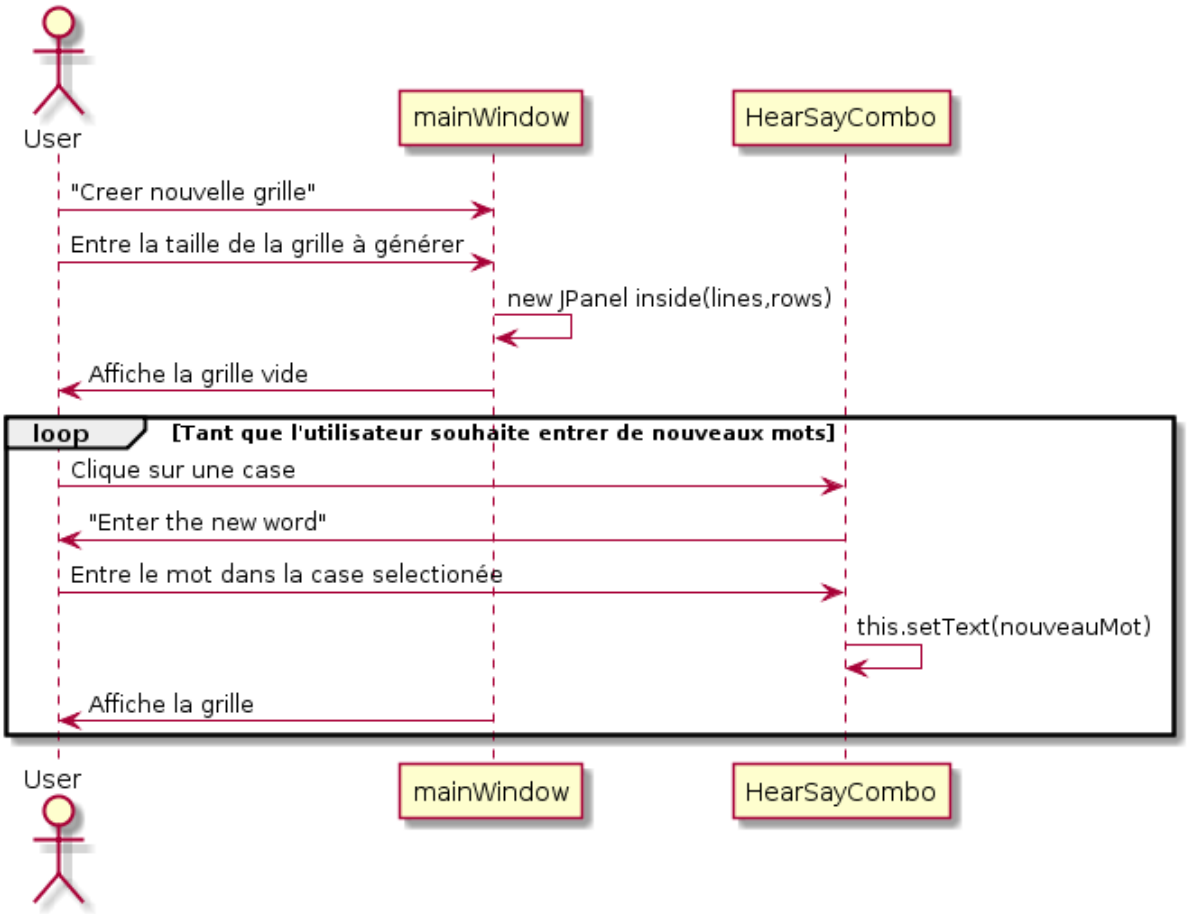


Figure 10 : Diagramme de séquence

Ceci est le diagramme de séquence pour le cas d'utilisation « Créer nouvelle grille ».

Le fonctionnement est le suivant : lors de la création d'une nouvelle grille, l'utilisateur devra entrer la taille de la grille à générer, la classe de la fenêtre principale (mainWindow) créera un nouveau JPanel qu'il remplira de plusieurs HearSayCombo (qui sont constitué notamment de deux JButton) qui constitueront les éléments du tableau (new JPanel inside(...)).

L'utilisateur devra cliquer sur les cases « I say » (de type JButton) qui ouvriront une fenêtre demandant d'entrer le nouveau mot, et la case se modifiera (this.setText(…)).

b. Diagramme de classes participantes

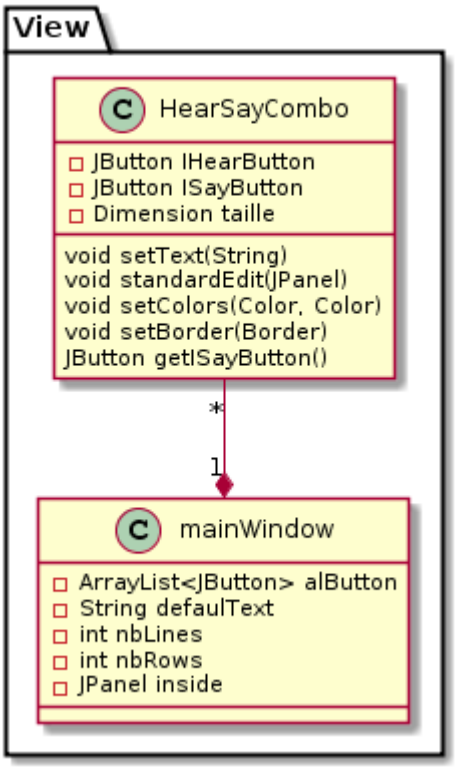


Figure 11 : Diagramme de classes participantes

Ici nous avons décrites les deux classes principales de l'application avec les méthodes appelées dans le diagramme de séquence ci-dessus.

Architecture MVC

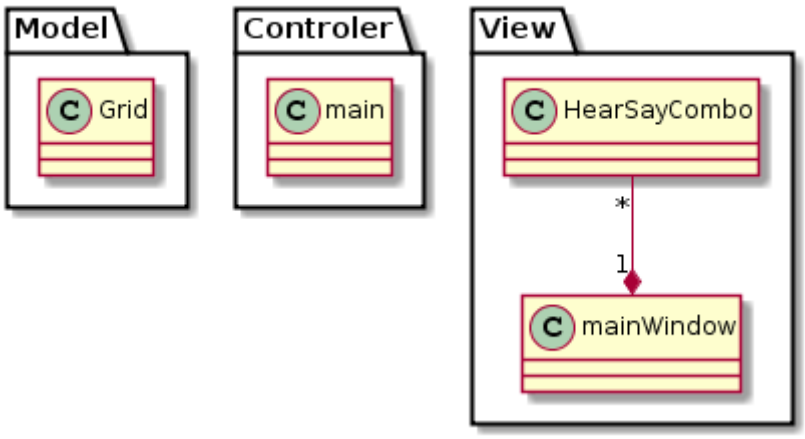


Figure 12 : Architecture MVC

Ceci est l'architecture MVC que nous avons mis en place pour l'implémentation de nos classes.

Dans la package View se trouve les deux classes comportant l'interface avec laquelle l'utilisateur interagit. La classe mainWindow étant la fenêtre principale contenant tous les boutons ainsi que la grille de HearSayCombo, qui est une classe associant deux JButton.

Dans le package Grid se trouve les méthodes pour mélanger et assembler les chaines de caractères, elles serviront aussi à constituer une seule et même chaine de texte, contenant tous les mots pour sauvegarder dans un fichier.

Quant au package Model, il contient la classe main, c'est notre classe « launcher », ne contient que la méthode main() qui sert à lancer l'application .

IV. Spécificités techniques

Afin de mette en place, nous allons développer une application Java. Cela demandera donc l'installation du logiciel *Java* chez l'utilisateur de l'application. Ce logiciel peut se télécharger gratuitement à l'adresse suivante :

https://www.java.com/fr/

Dans un premier temps l'utilisateur nécessitera également un logiciel permettant de lire et d'imprimer des fichiers de format csv, tel que *Microsoft Office Exel*, ou *Libre Office Calc.* Dans un second temps, la fonctionnalité « imprimer » sera implémentée.

Pour réaliser cette application, nous utiliserons l'environnement *Eclipse* pour son aide très précieuse durant l'écriture du code. Aussi, nous utiliserons les librairies « java AWT/Swing» qui offrent de nombreuses fonctionnalités pour créer une interface.

Le partage du travail au sein du groupe se fera grâce à *GitHub* qui permet une mise en commun des données, notamment du code. L'application offre également d'autres fonctionnalités très appréciables pour le développement, tel que le *versionning* des fichiers, ou encore différentes branches pour ranger les informations que nous partageons.

La communication au sein de l'équipe se fera grâce à *Facebook*/*Messenger*, ainsi que *Skype* qui offre une fonctionnalité de partage d'écran.

Enfin la réalisation de la documentation se fera grâce aux différents logiciels du pack *Microsoft Office*.

Nous utilisons PlantUML afin de générer nos diagrammes tels que « Diagramme de classe », « diagramme de séquence »…

V. Plan d’assurance qualité

Pour s’assurer de la pertinence de cette application, nous avons choisis de mettre en place la méthode agile. Cette méthode vise à réaliser régulièrement des rendez-vous avec le client et le superviseur afin d’obtenir un maximum de retour sur le projet et son avancement. De cette manière, nous nous assurons que l’application que nous développons répond bien aux exigences du client.

Aussi, nous envisageons le fait de pouvoir partager notre application avec quelques professeurs et quelques élèves afin qu’ils puissent nous faire des retours constructifs nous permettant d’améliorer notre rendu.

Nous implémenterons régulièrement des tests afin de garantir le bon fonctionnement des fonctionnalités implémentées.

VI. Organisation mise en œuvre dans l'étape DAC

Concernant la communication durant l'étape DAC, elle a été réalisée par plusieurs moyens. Tout d'abord au sein de l'équipe, grâce à plusieurs outils à savoir, Messenger, Skype, GitHub, réunion à l'IUT. Toutes nos données ont été échangés via la plateforme GitHub, notre projet est en ligne afin qu'on puisse y accéder depuis n'importe quel PC.

Dans un second temps, la communication avec le client et le superviseur à majoritairement été faite par mail et mais aussi grâce à des réunions à l'IUT. Il y a eu plusieurs réunions :

- Jeudi 22 Septembre 2016 (client)
- Jeudi 29 Septembre 2016 (superviseur + client)
- Lundi 17 Octobre 2016 (superviseur)
- Mercredi 07 Décembre 2016 (client)
- Une autre réunion a eu lieu (une avec le client)

Gantt prévisionnel et description

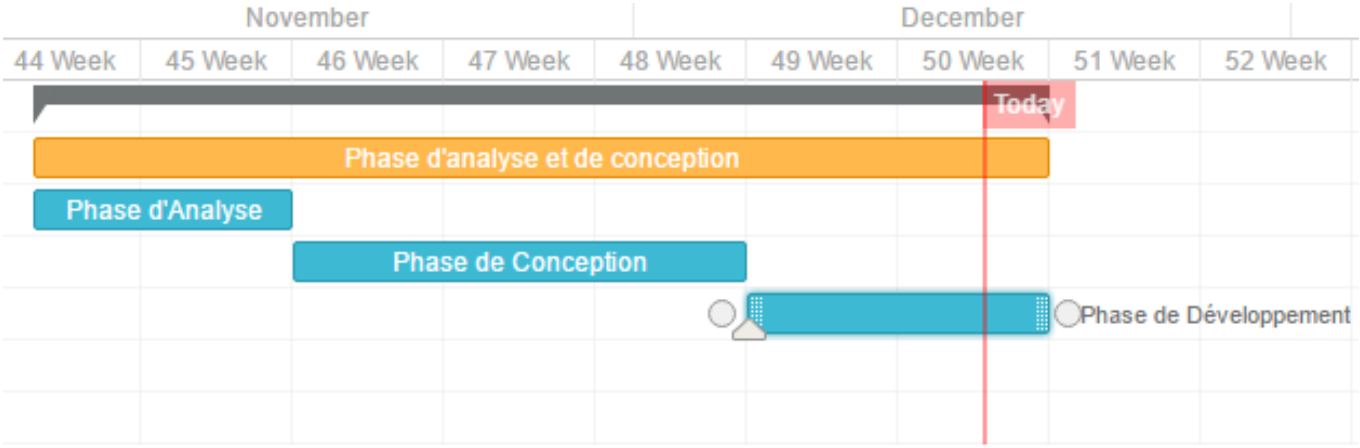


Figure 13 : Diagramme de Gantt prévisionnel

Lors de notre organisation prévisionnelle, nous avons choisi de diviser la phase d'analyse et conception en 3 sous-phases :

- Une première phase d'analyse ayant pour but de rechercher les solutions et les technologies les plus adaptées pour répondre aux besoins du client. Elle comprend aussi la mise en place des outils techniques et les outils de communication d'équipe.

- Une deuxième phase de conception pour structurer le projet, produire des diagrammes qui nous serviront à mieux organiser notre travail afin de rester efficace et pertinent. Elle servira de base et facilitera grandement la phase de développement.

- Une dernière phase de développement pour produire un prototype d'application qui suivra les choix de conception et qui répondra aux besoins spécifiés par le client dans le cahier des charges.

Nous avions prévu de garder un contact régulier avec le client afin d'avoir un suivi et des retours continus par celui afin que l'application répondent le plus précisément possible à ses besoins.

Gantt réel et décalages

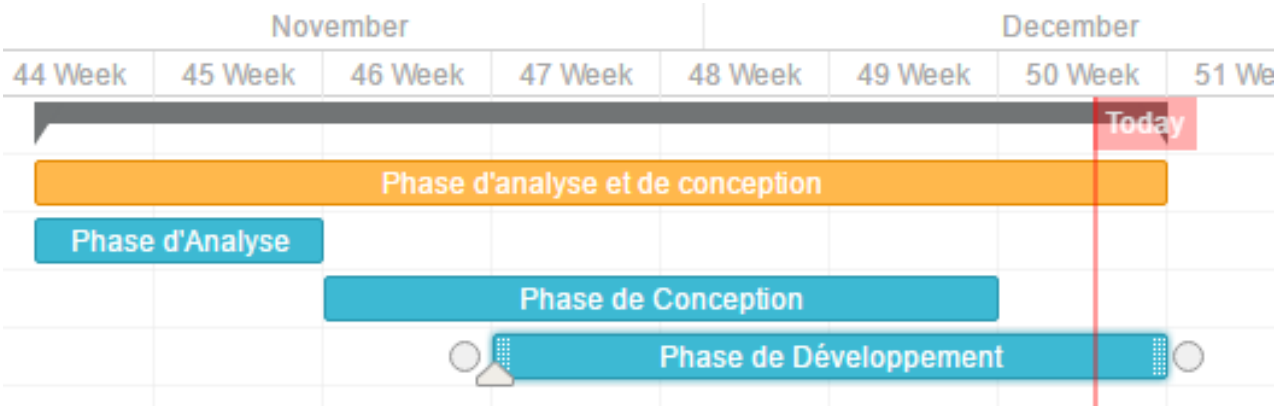


Figure 14 : Diagramme de Gantt réel

Ceci est le diagramme de Gantt du réel. Ils comportent deux décalages par rapport au Gantt prévisionnel.

En effet la phase de conception s'est terminée deux semaines plus tard, car nous avons eu des problèmes à choisir la bonne structure pour nos classes, notamment pour le stockage de données dans la classe Grid. Cela nous a augmenté la charge de travail.

En réponse à cela, nous avons anticipé et avons commencé la phase de développement en avance, en commençant notamment par développer les parties de l'application non dépendantes de la conception (interface).

Conclusion

L'un des enjeux principaux de ce sujet est de faciliter la production d'une activité pour notre client. En proposant une application plus ciblée sur les besoins du client nous espérons permettre à celui-ci de gagner du temps. Nous avons donc décidé de créer une application en Java permettant de répondre à ses besoins de manière efficace. L'application doit être le plus ergonomique possible et doit permettre une modification des grilles d'une manière très simple.

Annexes

I. Compte-rendu P-TUT 29/09/16 « I Hear, I Say »

Groupe M

Personnes présentes : - Isabelle Clavel

- Laurence Redon
- Guillaume Robert
- Yoann Gathignol
- Titouan Bouëte-Giraud

Sujet de la réunion : Discussion à propos des fonctionnalités du projet

Lors de notre dernière réunion nous avons ajouté les besoins suivants :

- L'implémentation d'un historique qui garde une trace des anciennes grilles générées, contenant également le nom d'utilisateur, le groupe, la date et la thématique de la grille,
- Filtrer l'historique pour retrouver les grilles.

Avant le rendu final nous devons rendre une version finale du cahier des charges au professeur référent. Le projet pourra potentiellement se terminer en avance, c'est pourquoi il pourrait y avoir des modifications. Aussi, il faudra expliquer la structure de notre algorithme.

II. Compte-rendu 07/12/2016 « I Hear, I Say »

Groupe M

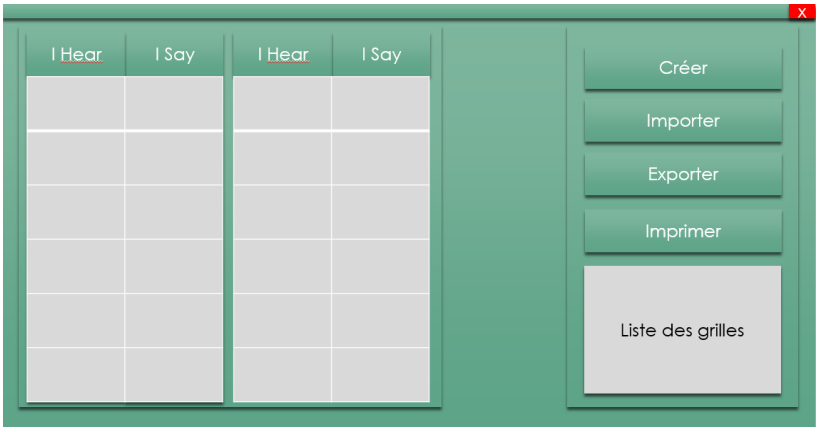
Personnes présentes : - Isabelle Clavel

- Guillaume Robert
- Yoann Gathignol
- Titouan Bouëte-Giraud

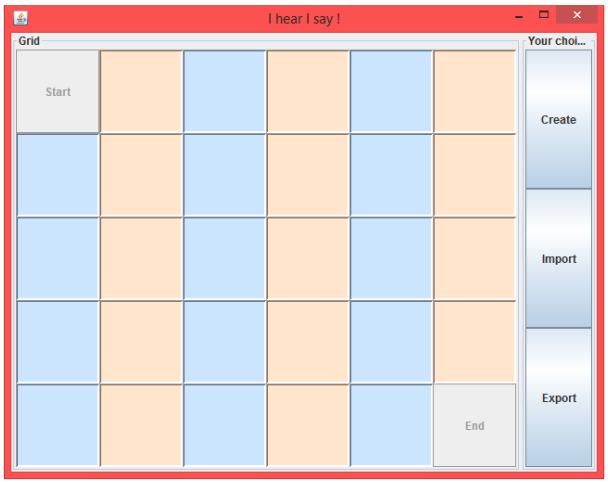
Sujet de la réunion : Présentation d'un premier prototype

Sujets évoqués :

- Présentation de la maquette



- Présentation de l'existant



Colonne bleu : I hear
Colonne beige : I say
Paire de boutons bleu/beige (même contenu)

- Un format type de csv sera fourni (ce format devra être respecté pour l'importation).
- Présente d'un dossier « Importation Automatiques » pour charger des grilles au lancement de l'application.
- Nous avons aussi discuté de la possibilité d'entrer tous les mots d'un coup sans avoir à cliquer sur les cases.