Julie Leclerc

Virgil Lacondemine

Laetitia Bourdon

Elodie Montcarmel

Hugo Leprunier

My Tomorrow

1. Délivrables
2. Délivrable 0

Composition du groupe :

Le groupe est composé de cinq élèves, des tâches spécifiques leur seront attribuées. Ces rôles sont attribués de manière indicative et seront probablement amenés à être modifiés.

* Hugo Leprunier : *Chef de projet / directeur fonctionnalités*
* Julie Leclerc : *développeur / directrice graphique*
* Elodie Montcarmel : *développeur*
* Laetitia Bourdon : *développeur / graphique*
* Virgil Lacondemine : *développeur / Directeur algorithmique*

Description du projet :

**Idée directrice :** Notre client rencontre des problèmes de gestion de planning dans le cadre professionnel. Le projet portera donc sur une gestion automatique d’agenda accompagné de différentes fonctionnalités.

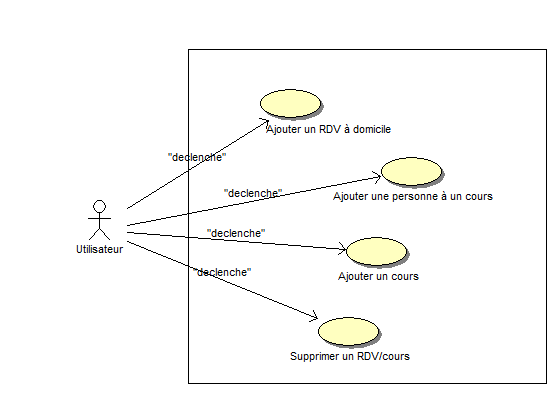
**Résultat final attendu :** Le rendu final du projet devra répondre aux attentes du client : organiser automatiquement un agenda. L’application devra être capable de trier différents objets (créneaux horaire, rendez-vous, jours,…) de manière à construire un planning. Celui-ci devra être fait sous des contraintes de priorités et de plages horaires restreintes (correspondant aux horaires de travail). Par exemple, le client doit passer 4h sur telle tâche et prévoit un rendez-vous le mardi à 11h telle semaine, l’application prendrait en compte ces contraintes et les tâches déjà présentes pour le client pour compléter son emploi du temps. Grâce à cette application, le client pourra chaque semaine sortir un planning personnalisé. Le but de cette application est donc d’optimiser le temps de travail en planifiant les semaines suivant les tâches à réaliser ainsi que les différents rendez-vous et réunions à effectuer.

**Perspectives :** L’application pourra être complétée par différentes fonctionnalités. La gestion d’alerte client/professionnel ou encore des aménagements d’horaires particuliers (ne pas dépasser 14h le mardi et centrer le travail aux alentours de midi le jeudi par exemple) seront des perspectives envisageables lors de la conception du projet. L’application pourrait aussi évoluer en prenant en charge différents employés et en réalisant un planning personnalisé pour chacun. Il prendrait ainsi en compte leur statut pour définir les différents horaires.

1. Délivrable 1

Notre projet se base sur les différentes demandes de notre client. En effet, notre cliente étant une sage-femme, elle alterne au quotidien entre des rendez-vous à domicile et des cours donnés à son cabinet (yoga, cours de respiration, de piscine …). On observe donc trois cas différents de création de tâches : créer un rendez-vous, un cours et ajouter une personne à un cours. Ce sont ces cas que nous détailleront ci-après. Comme pour toute application de planification d’agenda, il faut pouvoir supprimer des rendez-vous et des cours de l’emploi du temps. Cela constituera notre dernier cas d’utilisation.

## Diagramme de cas



## Documentation des cas

* Cas Ajout d’un rendez-vous à domicile

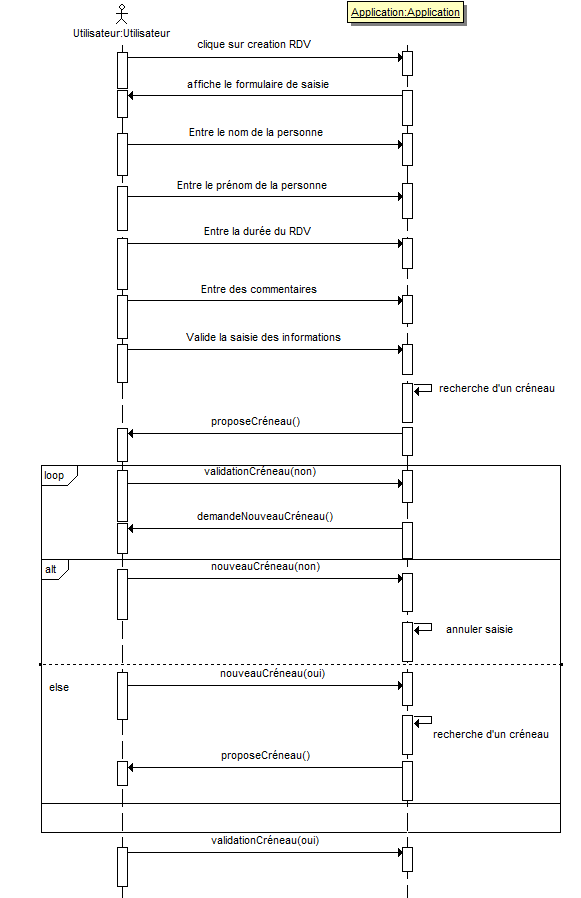
Début : nouveau rendez-vous à saisir  
Fin : le rendez-vous est saisi  
Paramètres : - Nom  
 - Prénom  
 - Durée du rendez-vous  
 - Commentaires  
Scénario : L’utilisateur de l’application (donc notre cliente) va entrer toutes ces informations dans l’application. Celle-ci lui proposera un créneau libre de son emploi du temps. Si celui-ci ne convient pas à la personne, la personne pourra choisir d’arrêter la prise de rendez-vous ou d’obtenir un nouveau créneau et ce, jusqu’à l’obtention d’un créneau qui lui convient.

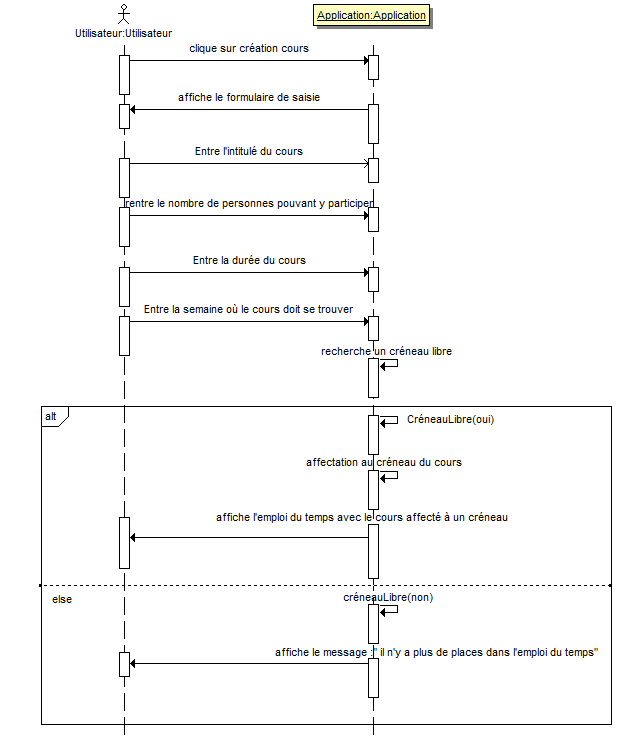
Diagramme de séquence :

* Cas Ajout d’un cours

Début : nouveau cours à saisir  
Fin : le cours apparaît dans l’emploi du temps  
Paramètres : - Intitulé du cours  
 - Nombre de personnes pouvant participer au cours  
 - Durée du cours  
 - Semaine où le cours doit se trouver

Scénario : L’utilisateur de l’application va entrer toutes ces informations dans l’application. Celle-ci regardera le planning pour voir s’il reste un créneau de libre pour placer le cours. S’il en existe alors l’application renvoie le planning avec le nouveau cours ajouté. Au contraire, s’il ne reste aucun créneau, l’application renvoie un message.

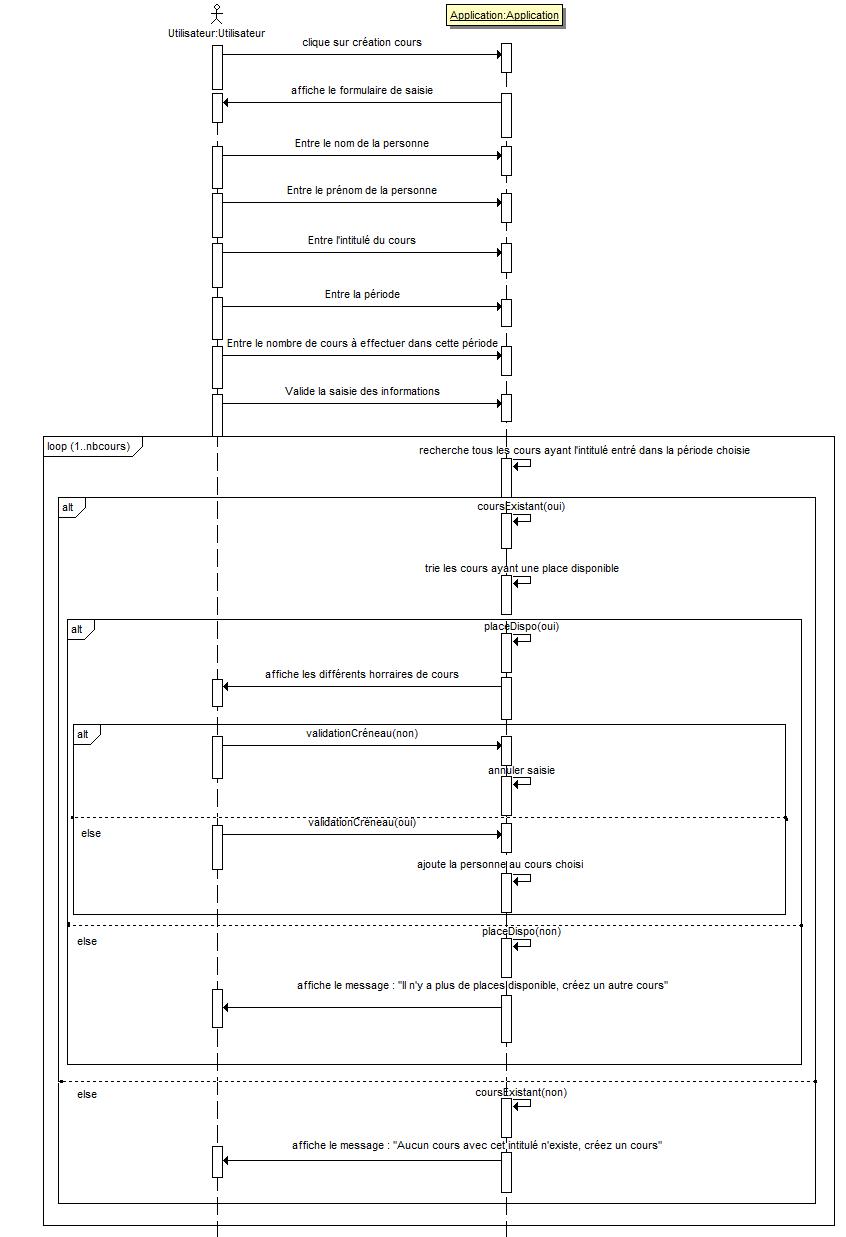
Diagramme de séquence :



* Cas Ajout d’une personne à un cours

Début : nouvelle personne à saisir dans un cours  
Fin : la personne est saisie dans un cours  
Paramètres : - Nom  
 - Prénom  
 - Intitulé du cours  
 - Période  
 - Nombre de cours à effectuer dans cette période  
Scénario : L’utilisateur de l’application va entrer toutes ces informations dans l’application. Celle-ci regardera le planning pour voir dans un premier temps si un ou plusieurs cours existent à cette période-là. S’il en existe, l’application affichera les créneaux horaires des cours ayant encore des places pour que l’utilisateur puisse choisir son créneau. Si aucun de ces horaires ne convient à la personne la saisie sera alors annulée (l’application ne créera pas d’autres cours ayant cet intitulé tant que ceux déjà placés ne sont pas tous remplis). Si au contraire, tous les cours sont remplis ou s’il n’existe pas de cours ayant l’intitulé saisi, alors l’application renverra à l’utilisateur un message lui demandant de créer un nouveau cours avant de vouloir ajouter une personne.

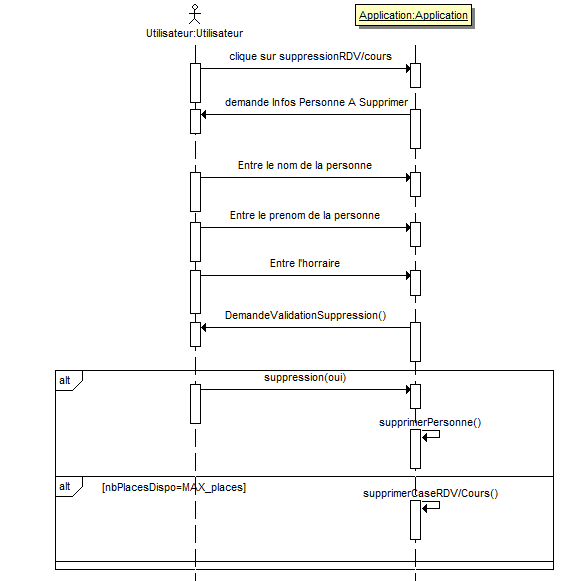
Diagramme de séquence : (voir page suivante)



* Cas Suppression d’un RDV/cours

Début : rendez-vous ou personne dans un cours à supprimer  
Fin : le rendez-vous ou la personne d’un cours est supprimé  
Paramètres : - Nom  
 - Prénom  
 - Horaire  
Scénario : L’utilisateur de l’application va entrer ces informations dans l’application. Celle-ci trouvera la personne dans l’emploi du temps à l’horaire saisi et lui demandera une confirmation pour supprimer le rendez-vous correspondant. L’application testera ensuite si le nombre de places disponibles restantes est égal au nombre de places maximum d’un rendez-vous ou d’un cours (on part du principe qu’il peut y avoir qu’une personne pour un rendez-vous et plusieurs pour un cours). On souhaite ainsi éliminer de l’emploi du temps les cases inutiles ne contenant plus aucune personne pour ainsi libérer un créneau.

Diagramme de séquence :

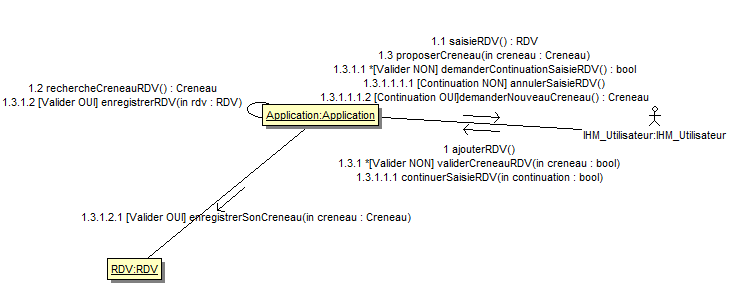


1. Délivrable 2

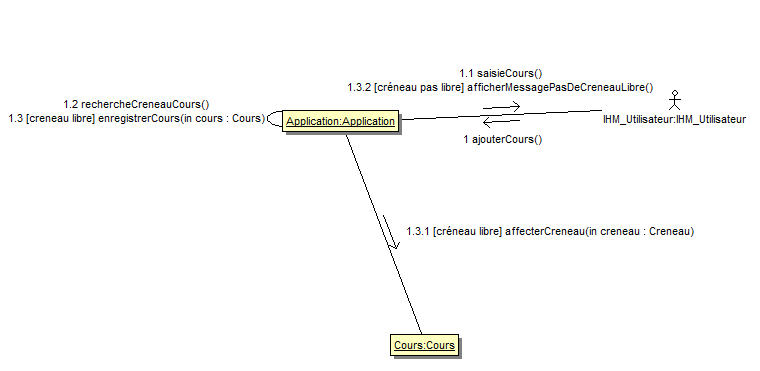
Notre projet se base sur les différentes demandes de notre client. En effet, notre cliente étant une sage-femme, elle alterne au quotidien entre des rendez-vous à domicile et des cours donnés à son cabinet (yoga, cours de respiration, de piscine …). On observe donc trois cas différents de création de tâches : créer un rendez-vous, un cours et ajouter une personne à un cours. Comme pour toute application de planification d’agenda, il faut pouvoir supprimer des rendez-vous et des cours de l’emploi du temps. Cela constituera notre dernier cas d’utilisation.

* Cas Ajout d’un rendez-vous à domicile

Diagramme de collaboration :

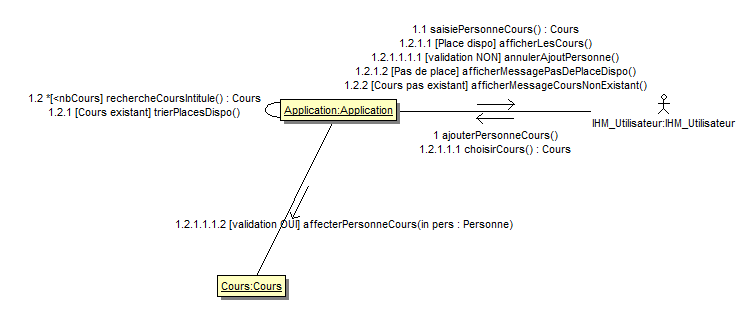


* Cas Ajout d’un cours

Diagramme de collaboration :

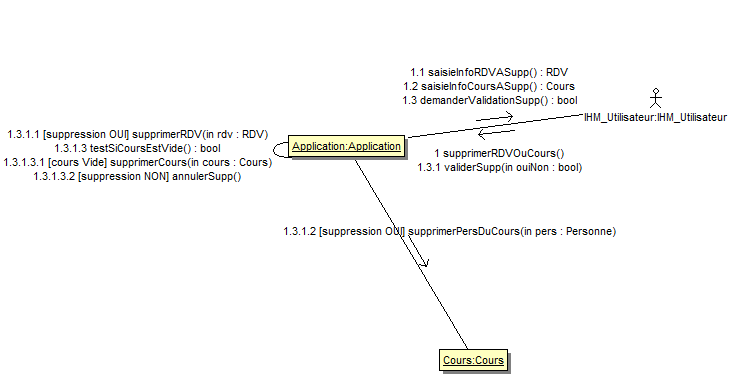
* Cas Ajout d’une personne à un cours

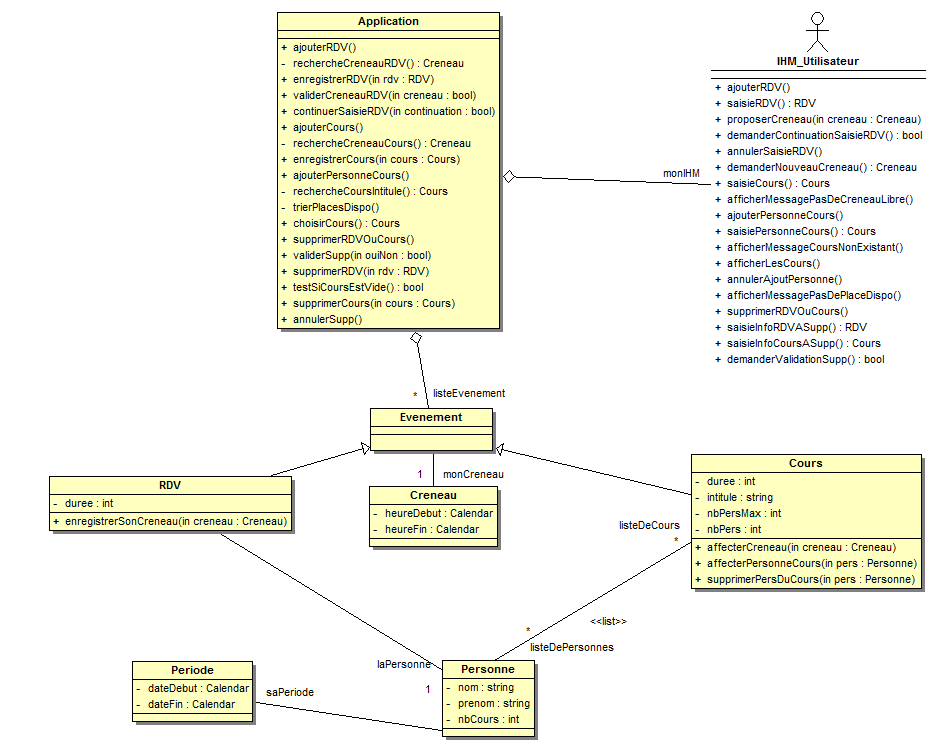
Diagramme de collaboration :



* Cas Suppression d’un RDV/cours

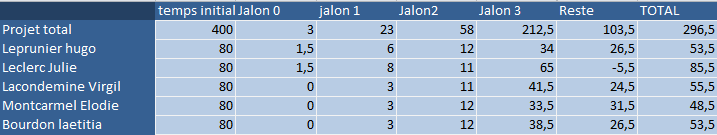
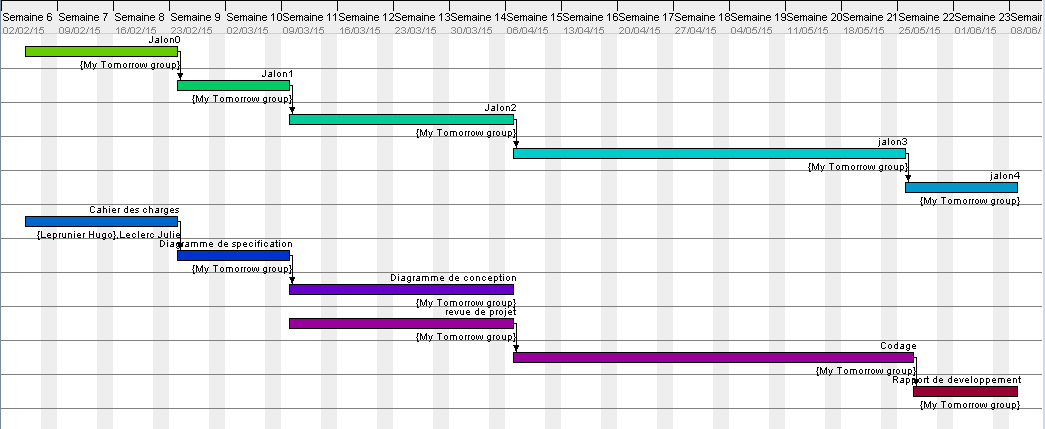
Diagramme de collaboration :



Diagramme de Classes :

1. Gestion de projet

Bilan Heures passées / Heures prévues



Répartition des taches

Les taches n’ont pas été clairement reparties. Pour des raisons de compréhension et d’apprentissage, l’ensemble du groupe a préféré travailler en commun de manière à ce que tous les membres intègrent certaines notions sans retard.

Certaines parties du code ont tout de même été réalisées de manière découpée de façon à gagner du temps de travail. Mais cette répartition n’était pas stricte.

Conduite de projet

Un diagramme de Gantt a été réalisé de manière à obtenir un suivi explicite des taches à réaliser.

Une réunion entre les membres du groupe a été faite après et avant chaque nouvelle période de jalon. C’est réunion ont permis de faire un briefing sur les tache à effectuer en priorité et sur la répartition du travail.

D’autres réunions « d’urgence » ont été planifiées lorsque le groupe était à un point critique (écartement du sujet, blocage, …).

Un planning a été fait pour l’ensemble du groupe : un créneau de travail les jeudis après-midi.

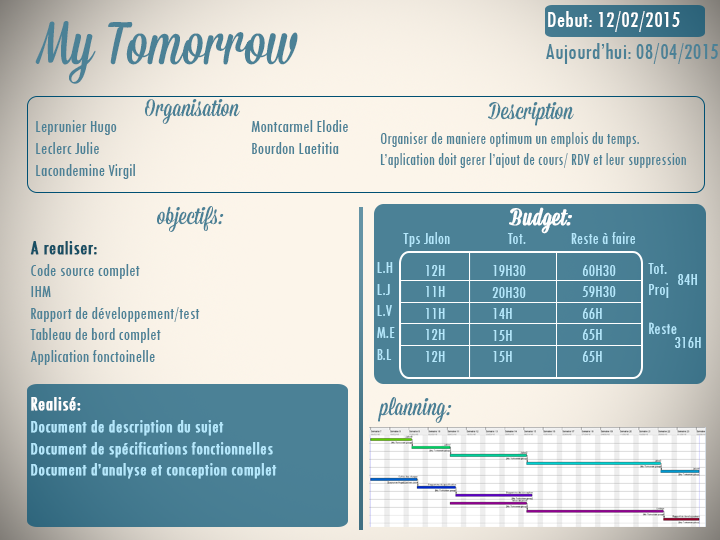
Des visio-conférences ont également été pratiquées de manière à combler les lacunes des jeudis après-midi et favoriser le contact à distance.

Carnet de bord :

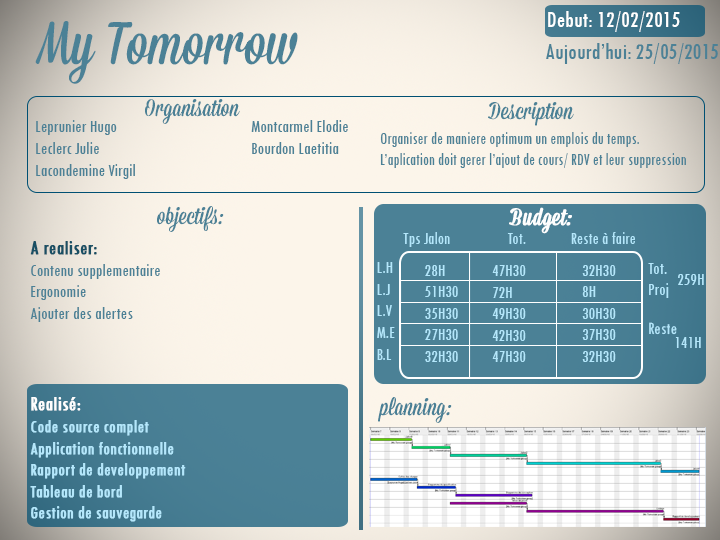
Jalon 1 :



Jalon  2 :



Jalon 3 :



1. Rapport de développement

Nos objectifs étaient de créer une application permettant de gérer le planning d’une personne. L’application devra être capable de trier différents objets (créneaux horaire, rendez-vous, jours,…) de manière à construire un emploi du temps avec le minimum de pause entre chaque cours ou rendez-vous.

Grâce à cette application, le client pourra chaque semaine sortir un planning personnalisé. Le but de cette application est donc d’optimiser le temps de travail en planifiant les semaines suivant les tâches à réaliser ainsi que les différents rendez-vous et cours à effectuer.

Ce qui a été fait :

Nous avons traité les 4 cas prévus au départ.

• Le premier cas est d’ajouter un rendez-vous. L’application nous demande le nom et le prénom de la personne qui souhaite obtenir un rendez-vous. Après validation, celle-ci demande le jour de disponibilité, puis la durée (en minutes) du rendez-vous.

Des créneaux sont alors proposés, nous pouvons choisir celui qui nous convient. Lorsqu’il n’y a plus de créneau, une fenêtre s’ouvre indiquant qu’il n’y a plus de créneau libre. Quand on valide le créneau, le rendez-vous se place automatiquement sur le calendrier, affichant le prénom et le nom de la personne ayant pris le rendez-vous.

Nous pouvons annuler à tous moment l’ajout d’un rendez-vous.

• Le deuxième cas est d’ajouter un cours. L’application demande l’intitulé du cours, le jour de disponibilité et, après validation, la durée (en minutes) du cours.

Des créneaux sont alors proposés, nous pouvons choisir celui qui nous convient. Lorsqu’il n’y a plus de créneau, une fenêtre s’ouvre indiquant qu’il n’y a plus de créneau libre. Quand on valide le créneau, le cours se place automatiquement sur le calendrier, affichant l’intitulé du cours.

Nous pouvons annuler à tous moment l’ajout d’un cours.

Pour ces deux cas, nous avons décidé d’afficher les créneaux sur l’IHM, sous forme de boutons. Lorsque l’on clique dessus, le détail du rendez-vous ou du cours apparaît dans une fenêtre.

Pour les cours, les personnes participantes sont affichées.

Nous pouvons également supprimer le cours ou le rendez-vous depuis cette fenêtre.

• Le troisième cas est l’ajout d’une personne à un cours. L’application demande le nom puis le prénom de la personne que nous voulons ajouter à un cours. Celle-ci demande la période où le client est disponible (jour de début/jour de fin) puis de saisir l’intitulé du cours auquel nous voulons l’ajouter.

S’il n’y a pas de cours crées avec cet intitulé, une fenêtre s’ouvre disant qu’il faut d’abord créer un cours.

S’il n’y a plus de place disponible dans un cours, le créneau du cours n’est alors plus proposé.

• Le quatrième et dernier cas est celui de la suppression. L’application demande le jour puis l’heure de début de l’évènement à supprimer.

Si l’évènement existe, une fenêtre s’ouvre indiquant que l’évènement va être supprimé.

Si l’évènement n’existe pas, une fenêtre s’ouvre indiquant qu’il n’y a pas d’évènement existant à cette date.

Nous avons également aménagé des horaires, il ne peut y avoir d’évènement avant 8h, entre 12h et 14h et après 18h.

Limites de l’application :

Lorsque l’application donne tous les créneaux disponibles d’une demi-journée (lorsqu’elle est vide), la liste des créneaux est exhaustive. En effet, si on souhaite créer un évènement d’une heure et qu’aucun évènement n’est planifié le matin. Il proposera un créneau de 8h à 9h puis de 9h à 10h et ainsi de suite. Il n’y aura donc pas possibilité de commencé l’évènement à 8h30 par exemple.

L’application n’est pas adaptable pour tous les utilisateurs. Elle s’adresse à des utilisateurs spécifiques c’est-à-dire ceux qui souhaitent ne pas programmer d’évènement entre 12h et 14h. Cette application s’adresse aussi à ceux voulant planifier des cours et des rendez-vous. Cependant ces paramètres étaient définis dans le cahier des charges comme l’application répond à un client potentiel.

Ce que nous pourrions rajouter pour améliorer notre application :

• Rendre l’application plus ergonomique (responsive design) car nous ne pouvons changer la taille de la fenêtre par soucis d’affichage du calendrier.

• Ajouter des alertes pour rappeler à l’utilisateur qu’un rendez-vous ou un cours allait bientôt avoir lieu.

• Faire un récapitulatif des personnes, lorsque l’on clique sur la personne participant au cours, nous voudrions pouvoir voir son adresse et son numéro de téléphone en complément de son nom et son prénom.

• Faire une base de données au lieu d’enregistrer les données dans un fichier texte.

Ce qui est couvert par les tests :

Nous avons effectué les tests avec JUnit sur les classes du model afin de tester tous les cas pour vérifier l’implémentation des méthodes de base.

Tests réalisés :

Sur la classe Lesson :

Pour la méthode hasTheSameTitle() :

|  |  |  |
| --- | --- | --- |
| Titre de la leçon | Titre à comparer | Résultat du test attendu |
| "Base poo" | "base poo" | true |
| "BASE POO" | true |
| "Base POO" | true |
| "BASEPOO" | false |
| "POO" | false |
| "Coo" | false |

Pour la méthode hasFreePlace() :

|  |  |
| --- | --- |
| Nombre de personnes de la leçon | Résultat attendu |
| 0 | true |
| 1 | true |
| 2 | true |
| 3 | false |

Pour la méthode personIndex() :

|  |  |  |
| --- | --- | --- |
| Personnes dans la liste | Personne recherchée | Résultat du test attendu |
| Bourdon Laetitia  Montcarmel Elodie | Leprunier Hugo | -1 |
| Bourdon Laetitia | 0 |
| Montcarmel Elodie | 1 |

Sur la classe Person :

Pour la méthode equals() :

|  |  |  |
| --- | --- | --- |
| Personne sur laquelle on effectue le test | Personne testée | Résultat du test attendu |
| Lacondemine Virgil | Leclerc Julie | false |
| Virgil Lacondemine | false |
| Lacondemine Virgil | true |

Sur la classe ScheduledEvent :

Pour la méthode isBefore() :

|  |  |  |
| --- | --- | --- |
| Evènement 1  (Evènement planifié le …) | Evènement 2 | Résultat du test attendu sur evènement1.isBefore(évènement2) |
| 10/06/2015 de 8h à 9h | Le 10/06/2015 de 8h à 9h | false |
| Le 11/06/2015 de 8h à 9h | true |
| 11/06/2015 de 8h à 9h | Le 10/06/2015 de 8h à 9h | false |

Sur la classe TimeSlot :

Pour la méthode isBefore() :

|  |  |  |
| --- | --- | --- |
| Créneau 1 | Créneau 2 | Résultat du test attendu sur créneau1.isBefore(créneau2) |
| 10/06/2015 de 8h à 9h | Le 10/06/2015 de 8h à 9h | false |
| Le 11/06/2015 de 8h à 9h | true |
| 11/06/2015 de 8h à 9h | Le 10/06/2015 de 8h à 9h | false |

Après l’exécution de ces tests, nous avons pu remarquer que ces méthodes fonctionnent comme nous les avons définies. Par ailleurs, des jeux d’essais on été réalisés cpour tester les limites de notre application.

L’application est-elle utilisable ?

L’application est actuellement fonctionnelle et répond aux cas d’utilisations définis précédemment.