



Ecole Polytechnique de l'Université de Tours

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2021-2022

Solution web service pour la conception d'un planning collaboratif



POLYTECH[®]
TOURS

Entreprise

Delphi Technologies

Delphi
Technologies

Tuteur entreprise

Étudiant

Nicolas MAGNE (DI5)

Tuteur académique

Ameur SOUKHAL

3 avril 2022

Liste des intervenants

Entreprise

Delphi Technologies
9 Boulevard de l'Industrie
41000 Blois, FRANCE
www.delphi.com

Delphi
Technologies

Nom	Email	Qualité
Nicolas MAGNE	nicolas.magne-2@etu.univ-tours.fr	Étudiant DI5
Ameur SOUKHAL	ameur.soukhal@univ-tours.fr	Tuteur académique, Département Informatique
		Tuteur entreprise



Avertissement

Ce document a été rédigé par Nicolas MAGNE susnommé l'auteur.

L'entreprise Delphi Technologies est représentée par susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Ameer SOUKHAL susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Nicolas MAGNE, *Solution web service pour la conception d'un planning collaboratif*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={MAGNE, Nicolas},
  title={Solution web service pour la conception d'un planning collaboratif},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iii
1 Introduction	1
1 Contexte.....	1
2 Objectifs.....	2
3 Bases méthodologiques.....	2
2 Description générale	3
1 Environnement du projet	3
2 Caractéristiques des utilisateurs.....	3
3 Fonctionnalités du système	4
4 Structure générale du système.....	5
3 État de l’art / Veille technologique	6
4 Analyse et conception	7
1 Analyse	7
1.1 Règles de priorités.....	8
1.2 Hypothèses utilisées.....	9
1.3 Spécifications.....	10
2 Exemple	10

5	Mise en oeuvre	11
1	Partie front-end	11
1.1	Architecture	11
1.2	User eXperience (UX).....	13
1.3	Vues	14
2	Partie solveur	16
6	Bilan et conclusion	18
1	Bilan du semestre 9	18
2	Bilan du semestre 10.....	21
3	Bilan auto-critique.....	21
	Annexes	22
A	Planification, gestion de projet	23
1	Evolution du projet	23
B	Cahier de Spécifications	25
1	Solveur Algorithmique	25
1.1	Les classes	25
1.2	Fonctions de tri	26
1.3	Fonctions de calculs	26
1.4	Fonctions d’affichage.....	27
2	Spécifications non fonctionnelles	27
2.1	Contraintes de développement et conception	27
2.2	Contraintes de fonctionnement et d’exploitation.....	27
2.2.1	Performances	27
2.2.2	Contrôlabilité	28
C	Rapport d’avancement	29
D	Document d’installation	30
E	Document d’utilisation	31
F	Cahier de test	32
G	Bibliographie	33
H	Glossaire	34
I	Acronymes	35

Table des figures

2 Description générale

2.1	Diagramme des cas d'utilisation de l'application extrait du PRD de Mr. Pelloux-Prayer	4
2.2	Schéma general du système extrait du PRD de Mr. Pelloux-Prayer	5

5 Mise en oeuvre

5.1	Architecture de la partie front-end	12
5.2	Vue principale de l'application	14
5.3	Vue de modification d'un test	14
5.4	Vue d'options des tâches	15
5.5	Diagramme de classes du solveur	16
5.6	Pseudo-code de l'algorithme du solveur	17

6 Bilan et conclusion

6.1	Diagramme de Gantt réalisé au debut du projet.....	18
6.2	Diagramme de Gantt réalisé en fin de S9.....	19
6.3	Extrait du résultat du solveur partiel	20
6.4	Diagramme de Gantt réalisé en fin de S10.....	21

A Planification, gestion de projet

A.1	Le diagramme de Gantt prévisionnel S9	23
A.2	Le diagramme de Gantt prévisionnel S9-S10 mis a jour	24

C Rapport d'avancement

C.1 Rapport d'avancement	29
--------------------------------	----

F Cahier de test

F.1 Résultats des tests	32
-------------------------------	----

1

Introduction

Dans cette première partie nous présenterons le contexte du projet, ses objectifs, et pour finir ses bases méthodologiques.

1 Contexte

Ce projet s'inscrit dans le cadre de la formation d'ingénieur informatique à Polytech Tours de 5ème année. Il s'agit d'un **Projet de Recherche et Développement (PRD)** se déroulant du 17 septembre 2021 au 12 avril 2022. Le projet en lui-même est à la charge d'un seul étudiant et un enseignant chercheur est affecté à sa supervision. Ce PRD est réalisé par Nicolas MAGNE, étudiant en 5ème année à Polytech Tours et auteur de ce rapport, le tuteur académique associé est M. Soukhal.

Le client à l'origine du projet est l'entreprise Delphi technologies, groupe multinational américain spécialisé dans divers secteurs automobiles tels que les systèmes d'injection d'essence et de carburant diesel, les actionneurs ou les systèmes de transmissions. Cette entreprise compte aujourd'hui quelque 21.000 salariés (2018) et dispose d'un site de fabrication de système d'injection diesel à Blois. Ce sont donc plus précisément les équipes de ce site qui sont à l'origine de ce projet.

Ce projet a été étudié au préalable par des étudiants de la filière d'informatique industrielle à Polytech Tours dans le cadre d'un projet collectif d'octobre 2019 à juin 2020. L'équipe projet était composée de 5 étudiants Jean RENAUD, Pierre LE GALLIARD, Antoine BOURNERON, Célestin BOURCIER et Andrey PIVIDORI. Leurs missions étaient d'établir un cahier des spécifications, une analyse et modélisation des développements à réaliser (les documents sont disponibles en annexes) et un livrable fonctionnel ainsi qu'un livre de recettes pour valider le travail réalisé.

Il a été repris en 2020-2021 par Arthur Pelloux-Prayer étudiant à l'époque en 5ème année dans la filière informatique à Polytech Tours dans le cadre de son PRD, sa mission était de continuer le travail effectué par ses prédécesseurs. Il a particulièrement travaillé sur la partie **Back-End** avec le **Framework** Spring ainsi que sur la partie **Front-End** avec le framework VueJs.

2 Objectifs

Comme le mentionne mon prédécesseur Arthur Pelloux-Prayer dans son PRD[2] : le projet en lui-même consiste à créer une application pour faciliter la mise au point du planning des techniciens du centre de Blois. Concrètement, le site dispose d'un côté de trois bancs de tests de véhicules, d'un nombre donné de techniciens ayant des compétences propres, des périodes de disponibilités et de l'autre côté des clients ayant des disponibilités pour effectuer des tests sur leurs véhicules.

L'algorithme qui prend toutes ces entrées et produit un planning respectant au mieux les contraintes sera appelé le solveur dans la suite de ce rapport. Ce solveur est le cœur du projet, mais toutes les données d'entrées doivent pouvoir être consultées et modifiées via une interface web simple et conviviale.

3 Bases méthodologiques

Au total l'infrastructure du projet se compose :

- D'une partie Front-End permettant la saisie / modification / suppression des entrées du solveur, le lancement du solveur et l'affichage des résultats, à réaliser entièrement.
- D'une base de données pour faire persister les informations du système (les clients et leurs disponibilités, les employés, les plannings générés, les utilisateurs de l'application et leurs rôles), qui existe, mais doit être modifié.
- D'une partie Back-End avec une api permettant au Front-End d'interagir avec la base de données et le solveur, qui existe, mais doit être légèrement modifiée.
- D'une partie solveur qui sera chargée de calculer un emploi du temps en respectant les contraintes qui lui seront rentrées, à réaliser entièrement

L'infrastructure a été schématisée dans la section 2.2(Chapitre 2)

2

Description générale

1 Environnement du projet

L'existant au démarrage de ce projet est une partie Front-End qui doit-être refaite, une partie Back-End non complète et une base de données mal implémentée.

Je dois donc reprendre ces parties, c'est à dire :

- Développer le solveur qui n'existe pas encore.
- Refaire la partie **Front-End**.
- Ajouter des composants au **Back-End** afin de gérer les groupes de techniciens.
- Modifier la base de données pour ajouter des duos d'opérateurs.

Le travail réalisé par les étudiants précédents a été analysé et retranscrit (cf. Annexe **C**)

2 Caractéristiques des utilisateurs

Dans le système on distingue 3 types d'utilisateurs :

- L'administrateur, il peut :
 - Lire et écrire sur toutes les données du système (rôles, commentaires, données d'entrées, données de sorties...)
 - Lancer le solveur
 - Lire et écrire des commentaires publics ou privés
- L'opérateur, il peut :
 - Voir les tâches planifiées
 - Lire les résultats du solveur
 - Lire et écrire des commentaires publics
- Le client, il peut :
 - Voir les tâches planifiées
 - Lire les résultats du solveur
 - Lire et écrire des commentaires publics

Bien que le client et les opérateurs aient les mêmes droits sur l'application, la distinction est nécessaire pour l'écriture de commentaires et éviter les confusions.

3 Fonctionnalités du système

Les fonctionnalités du système ont été synthétisées sous forme d'un diagramme des cas d'utilisation ci-dessous

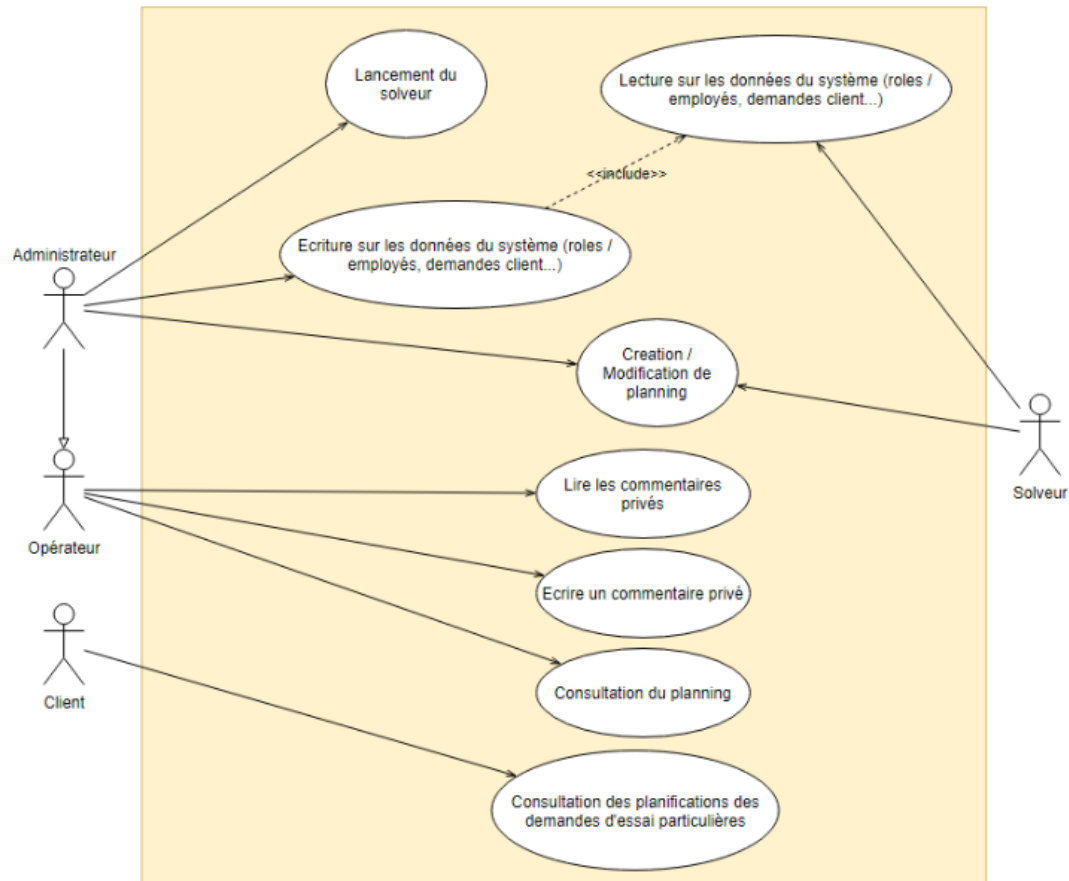


Figure 2.1 – Diagramme des cas d'utilisation de l'application extrait du PRD de Mr. Pelloux-Prayer

4 Structure générale du système

Les différentes composantes du système ont été schématisées ci-dessous

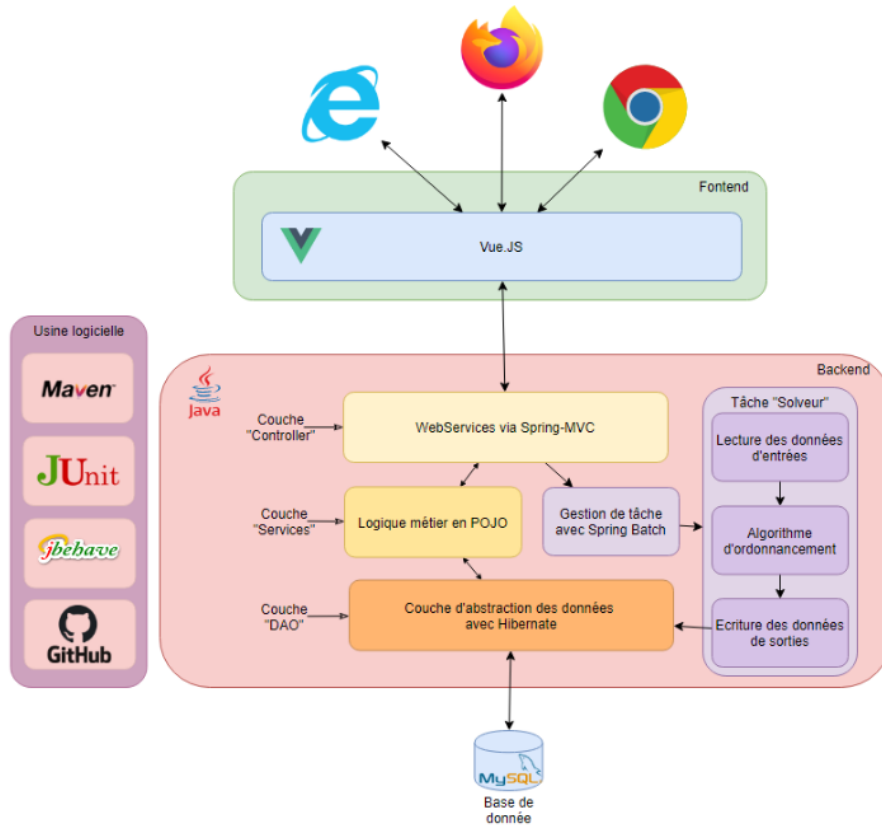


Figure 2.2 – Schéma general du système extrait du PRD de Mr. Pelloux-Prayer

3

État de l'art / Veille technologique

Un état de l'art très complet est présenté dans la thèse de Mr. Cheikh Mohamed DHIB (2013) intitulée 'Etude et résolution de problèmes d'ordonnancement de projets multi-compétences : intégration à un ERP libre'[1]. Il y est présenté différentes approches pour résoudre des problèmes d'ordonnancement de projet multi-compétences dont un en particulier le Preemptive Adapted Parallel Schedule Scheme Algorithm (PAPSSA) a servi de base algorithmique pour le solveur.

Cet algorithme renvoie un emploi du temps optimisé selon des règles appelées **Key Performance Indicator (KPI)** dans la suite de ce rapport. Pour les besoins du projet, de nouveaux KPI que ceux présentés dans la thèse ont dû être développés (cf. Chapitre 4). De plus l'algorithme a dû être modifié car la réalité du client ne permet pas la préemptivité des tâches.

4

Analyse et conception

1 Analyse

Une analyse de l'algorithme PAPSS à été réalisée par M. Pelloux-Prayer dans son PRD [2], il présente l'idée générale de l'algorithme PAPSS comme suit :

Un projet P défini par :

- Un ensemble de tâches à planifier A (plus deux fictives une de début et une de fin la première ayant une nécessité de commencer avant toutes les autres, la seconde devant commencer après toutes les autres)
- Un horizon de planification : un temps maximum pour l'ordonnancement du projet

Un ensemble de personnes Ps définis par :

- Un identifiant unique
- Une liste de compétences
- Une liste de période d'indisponibilité

Une règle de priorité R1 : qui va déterminer le tri initial de la liste des tâches, les différentes règles de priorités sont présentées dessous.

Pour chaque tâche du projet :

- Un identifiant unique
- Une date de début au plus tôt
- Un booléen indiquant si la tâche est préemptive
- Le nombre d'unités de temps requis

L'idée générale de l'algorithme est que pour chaque période de temps t entre 0 et l'horizon du projet, on récupère l'ensemble des tâches éligibles à l'instant t . Pour qu'une tâche soit éligible, elle doit remplir deux conditions :

- Ses prédécesseurs doivent être terminés
- Sa date de fin au plus tard doit être inférieure ou égale à t

Ces tâches sont ensuite triées selon la règle de priorité R1. On parcourt la liste dans l'ordre et on essaye d'ordonner cette tâche en déterminant les ressources qu'elle va utiliser grâce à l'heuristique d'affectation implémentée dans la méthode TrySchedule. Si aucune affectation n'est réalisable avec les ressources disponibles, on passera la tâche suivante. Une fois toutes les tâches

examinées, on passe à la période suivante qui correspond à la prochaine mise en disponibilité d'une ressource ou l'éligibilité d'une nouvelle tâche donnée par la méthode `nextEvent`.

Algorithme 1: Preemptive Adapted Parallel Schedule Scheme Algorithm(*PAPSS-R1*)

ENTRÉES: \mathcal{A} : l'ensemble de tâches à planifier

$t \leftarrow 0$

tantque $\mathcal{A} \neq \emptyset$ et $t < horizon$ **faire**

ES^t : ensemble de tâches éligibles à t , triées selon la règle de priorité $R1$

$i = 0$

tantque $i < |ES^t|$ **faire**

$feasible \leftarrow TrySchedule(ES^t(i), t)$

si $feasible$ **alors**

$\mathcal{A} \leftarrow \mathcal{A} \setminus ES^t(i)$

fin

$i \leftarrow i + 1$

fin tantque

$t \leftarrow nextEvent()$

fin tantque

1.1 Règles de priorités

Les différentes règles de priorités (KPI) sont définies dans la thèse [1] comme il suit :

1. Earliest Release Date (ERD) : une tâche A_i est plus prioritaire qu'une tâche A_j si leur date de début au plus tôt vérifie $r_i < r_j$
2. Earliest Due Date (EDD) : les tâches dont les dates de fin impératives (d_i) sont les plus petites sont placées en priorité
3. Minimum Slack (MINSLK) : les tâches sont triées dans l'ordre croissant de leur marge libre donnée par $\max(0, d_i - p_i - r_i)$
4. Most Number of Successors (MTS) : les tâches ayant le plus grand nombre de successeur sont placées en priorité
5. Most number of skills (MSSKILL) : les tâches demandant le plus grand nombre de compétences sont les plus prioritaires
6. Total Man Power Needed (MAXCHRG) : les tâches sont triées selon l'ordre décroissant de leur charge totale.
7. Criticality (CRITICITY) : les tâches les plus critiques sont prioritaires. Les tâches nécessitant les compétences les plus rares et les moins disponibles sont plus critiques.
8. Greatest rank (GRNK) : les tâches sont triées dans l'ordre décroissant selon la charge totale de leurs successeurs.
9. Greatest rank 2 (GRNK2) : les tâches sont triées dans l'ordre décroissant selon le nombre total de compétences requises par leurs successeurs.
10. Greatest rank 3 (GRNK3) : les tâches sont triées dans l'ordre décroissant selon la durée totale de leurs successeurs.
11. Random priority (RAND) : des priorités aléatoires sont associées aux tâches.

Algorithme 2: TrySchedule(tâche A_i , période t)

```

 $flow \leftarrow$  flot calculé en résolvant le problème d'affectation dans 3.4.1.3
si  $flow < |SK_i|$  alors
    retourner faux
fin si
fixer l'affectation de  $A_i$  en affectant à chaque compétence la personne dont un flot a
été établie entre son nœud et le nœud de la compétence
si  $A_i$  n'est pas préemptive alors
    planifier chaque compétence  $S_k$  de  $A_i$  de  $t$  à  $t + p_{i,k}$ 
sinon
    planifier chaque compétence  $S_k$  de  $A_i$  de  $t$  à  $t + 1$ 
    pour tout compétence  $S_k$  de la tâche  $A_i$  faire
        si la charge restante  $p_{i,k} - 1$  peut être planifiée alors
            planifier les  $p_{i,k} - 1$  périodes restantes de  $S_k$  le plus tôt possible selon la
            disponibilité de la personne affectée
        sinon
            retourner faux
    fin si
    fin pour
fin si
pour tout tâche  $A_j$  interrompue par  $A_i$  faire
    si les disponibilités des ressources permettent de reprendre  $A_j$  alors
        reprendre  $A_j$ 
    sinon
        retourner faux
    fin si
fin pour
retourner vrai

```

1.2 Hypothèses utilisées

Cependant cet algorithme doit être modifié car :

- Les tâches ne sont pas préemptives.
- Certaines contraintes du client ne sont modélisées.
- Les règles donnent lieu à des déséquilibres de charge de travail entre les techniciens.

Il est donc nécessaire de créer un nouveau KPI qui sera un compromis entre :

1. **Load Balancing (LB)** : L'opérateur qui sera chargé de la tâche sera celui parmi les opérateurs potentiels ayant le moins d'heures de travail à son actif.
2. **Earliest Due Date (EDD)** : Les tâches sont triées par ordre croissant selon la proximité de leurs dates de rendu.
3. **Closest Number of Skill (CNSKILL)** : Les opérateurs potentiels d'une tâche sont triés par ordre décroissant en fonction de la taille de l'intersection entre leur compétences et les compétences de la tâche. Sachant qu'ils doivent tous au minimum pouvoir remplir la tâche.

Aussi l'algorithme doit-être modifié car :

- La notion de précedence n'existe que très rarement pour des tâches de mises en route des machines.

- La notion de clients et de machines doit être ajoutée
- La notion de date de début au plus tôt d'une tâche n'existe pas
- Les tâches ne sont pas préemptives

1.3 Spécifications

Le solveur algorithmique développé s'articule autour de 6 classes :

- Un opérateur
- Une machine
- Un client
- Une tâche
- Des options du client
- Une entreprise contenant le tout

Pour les spécifications détaillées du solveur voir Annexe **B**

2 Exemple

Un exemple de scénario plutôt simple serait :

- Opérateurs : François et Jean ayant pour compétences respectives [1] et [1, 2, 3]
- Clients : Monsieur Jombel dont le véhicule nécessite les tâches dont les compétences sont les suivantes : [1] et [2, 3] (1h) et son fils dont le véhicule nécessite les tâches dont les compétences sont les suivantes : [1, 2, 3] et [1] (2h)
- François n'est pas disponible le Lundi de 8h à 10h

Lundi :

- 8h-9h : Jean réalise les tâches n°1 et 2 de M. Jombel car François ne travaille pas
- 9h-10h : Jean réalise la tâche n°1 du fils de M. Jombel car François ne travaille pas
- 10h-11h : François réalise la tâche n°2 du fils de M. Jombel car il a moins travaillé que Jean et peut réaliser la tâche

Si un opérateur peut réaliser la tâche avant un autre alors il réalisera la tâche (EDD)

Si deux opérateurs peuvent effectuer la même tâche en même temps, mais que le premier a moins travaillé que le deuxième, alors le premier réalisera la tâche. (LB)

Si plusieurs opérateurs peuvent effectuer la même tâche en même temps en sachant qu'ils ont tous autant travaillé alors celui dont les compétences correspondent le plus à la tâche sera choisie : par exemple les opérateurs ont les compétences suivantes : [1], [1,2,3,4] et [1,2] alors si la tâche ne nécessite que la compétence [1] alors le premier opérateur sera choisi (CNSKILL)

5

Mise en oeuvre

J'évoquerai dans cette partie les différentes architectures, décisions, et compétences utilisées pour développer la partie front-end dans un premier temps et la partie solveur algorithmique dans un second temps.

1 Partie front-end

Réaliser une partie front-end pour un outil aussi complexe n'est pas une tâche facile, souvent le développeur rencontre un dilemme entre faire une application facile à développer, mais pas pratique pour l'utilisateur ou faire une application complexe à développer, mais pratique pour l'utilisateur.

Dans mon cas j'ai préféré la seconde option, car c'est à mon sens la plus intéressante, aujourd'hui les applications complexes sont majoritairement facile d'utilisation. Le premier objectif c'est de savoir comment rendre ce type d'application facile à utiliser, c'est ce que nous verrons dans la première partie.

1.1 Architecture

Pour cette application j'ai décidé d'utiliser une architecture moderne appelée "Atome, Molecule, Organisme" (Atomic design). L'architecture **Modele, Vue, Controlleur (MVC)** est un standard aujourd'hui dans le développement, mais cette architecture n'est pas possible avec des Frameworks Javascript comme Vue.js ou React.js, car l'organisation d'un fichier comprend elle même une partie Vue et Controlleur qui sont donc insécables. La première architecture apparue pour ces langages est dite fonctionnelle. Chaque composant à une place dans l'application et le composant est développé autour de sa fonction dans une sous-partie de l'application.

L'approche Atomique est venue révolutionner ces architectures en proposant des composants réutilisables à l'infini dans toute l'application. Elle s'organise comme suit :

- Atome : composant personnalisé ne contenant aucun autre composant personnalisé.
- Molecule : composant personnalisé contenant des atomes.
- Organisme : composant personnalisé contenant des molécules.

Les modèles peuvent être placés dans des fichiers / dossiers différents. Voici l'architecture de l'application :

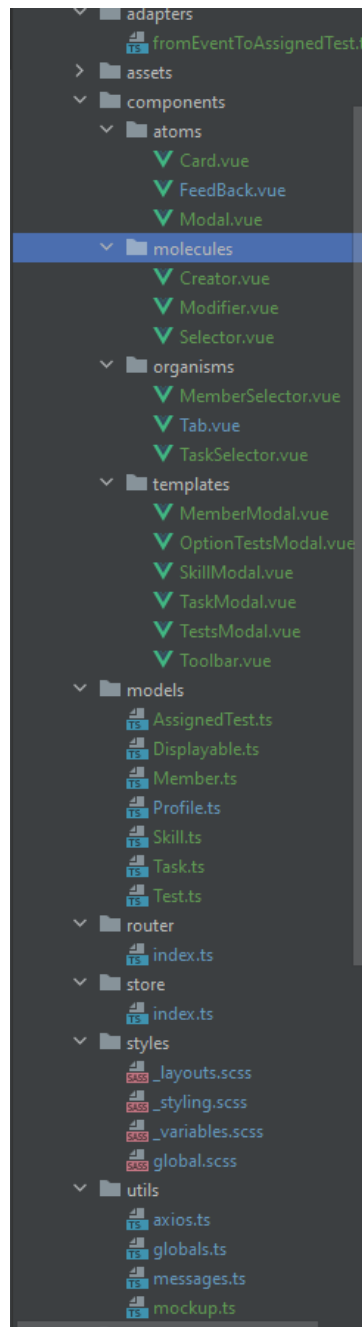


Figure 5.1 – *Architecture de la partie front-end*

Ainsi nous pouvons aussi retrouver une organisation des styles SCSS et des utilitaires comme le fichier de mockup qui permet de charger des données de test.

1.2 User eXperience (UX)

Les pratiques visant à assurer la satisfaction des utilisateurs d'une application peuvent être généralisées sous le terme d'**User eXperience (UX)**.

Dans le cadre de l'application, il a 3 leviers d'actions pour assurer une bonne **UX**.

- La simplicité de l'interface.
- La multiplication des chemins d'actions.
- La cohérence de l'interface face au sens commun.

La simplicité de l'interface : le nom est assez équivoque, une interface claire de sens, des actions nommées dont les noms sont compréhensibles de sorte que l'utilisateur comprend l'action qu'il est en train de réaliser ainsi que ces tenants et aboutissants.

Exemple de mauvaise UX : un bouton-icône dont le contexte de la page ne permet pas de prévoir l'action qui se réalisera lors d'un clique.

La multiplication des chemins d'actions : c'est le fait de prévoir plusieurs chemins différents pour réaliser la même action de sorte que de tout point de l'application l'utilisateur ne doivent pas revenir en arrière et perdre son avancement du fait qu'il est oublié d'effectuer une action au préalable.

Exemple de mauvaise UX : un utilisateur souhaite ajouter une tâche dans l'application, il prend quelques minutes pour renseigner tous les champs pour se rendre compte qu'au niveau du dernier champ permettant de choisir un opérateur, celui qu'il avait prévu n'existe pas encore dans l'application, il est donc obligé de fermer la fenêtre pour revenir sur la création d'un opérateur cela fait perdre du temps.

La cohérence de l'interface face au sens commun : ce levier a plusieurs interprétations possibles en fonction de la communauté d'utilisateurs qui se servent de l'application. Généralement il décrit le fait d'utiliser des standards intuitifs comparés aux standards actuels. Étendu à la communauté, il décrit le fait d'utiliser des notations, agencements, positionnements, couleurs reconnaissables facilement par la communauté du fait qu'elle les ai déjà vues ou utilisés.

Exemple de mauvaise UX : Afficher un emploi du temps qui ne correspond pas aux besoins du client et/ou qui ne respecte pas des règles établies.

Nous verrons maintenant de quelle manière j'ai utilisé ces leviers pour assurer une bonne UX avec quelques exemples de vues de l'application.

1.3 Vues

Cette partie ne présentera pas toutes le vues de l'application uniquement celles nécessaires a la présentation des exemples enoncés dans la sous-section précédente.

Voici la vue principale de l'application, elle sert de point d'entrée aux différentes actions ainsi que l'affichage de l'emploi du temps **Figure 5.2** :

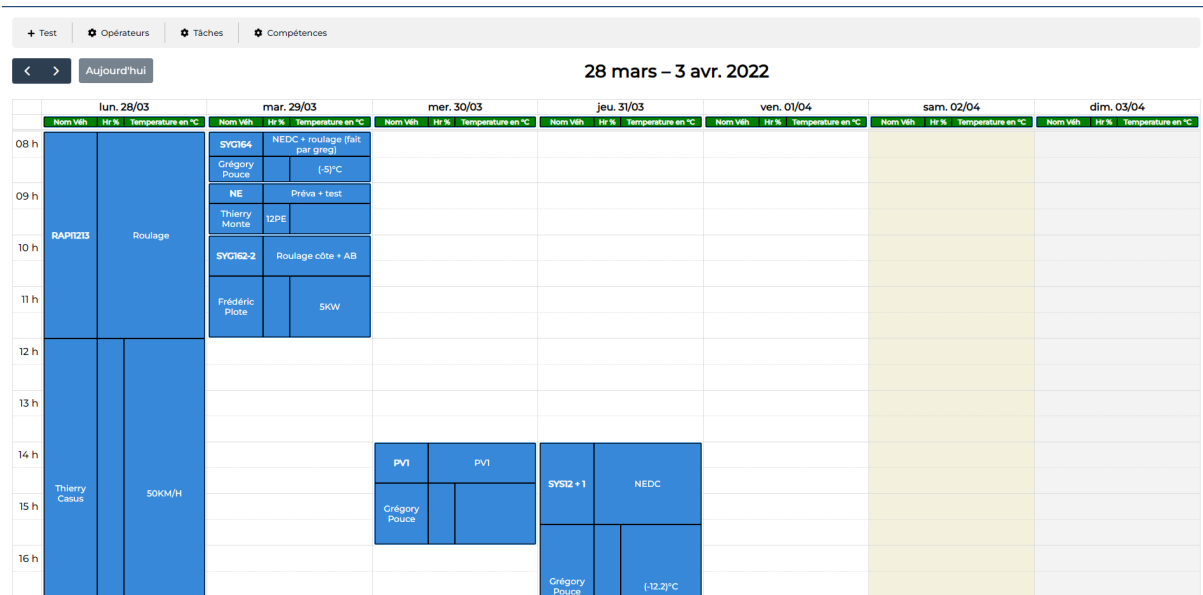


Figure 5.2 – Vue principale de l'application

Les boutons de la barre d'outils en haut de la vue sont nommés et titrés sans être trop disgracieux grâce à une icône, respectant le levier n°1 et n°3.

La partie affichage de l'emploi du temps respecte les standards déjà utilisés par les utilisateurs, en cliquant sur une période on peut modifier la modifier, respectant le levier n°1, n°2 et n°3.

Pour présenter la suite, voici la vue de modification d'un test **Figure 5.3**.

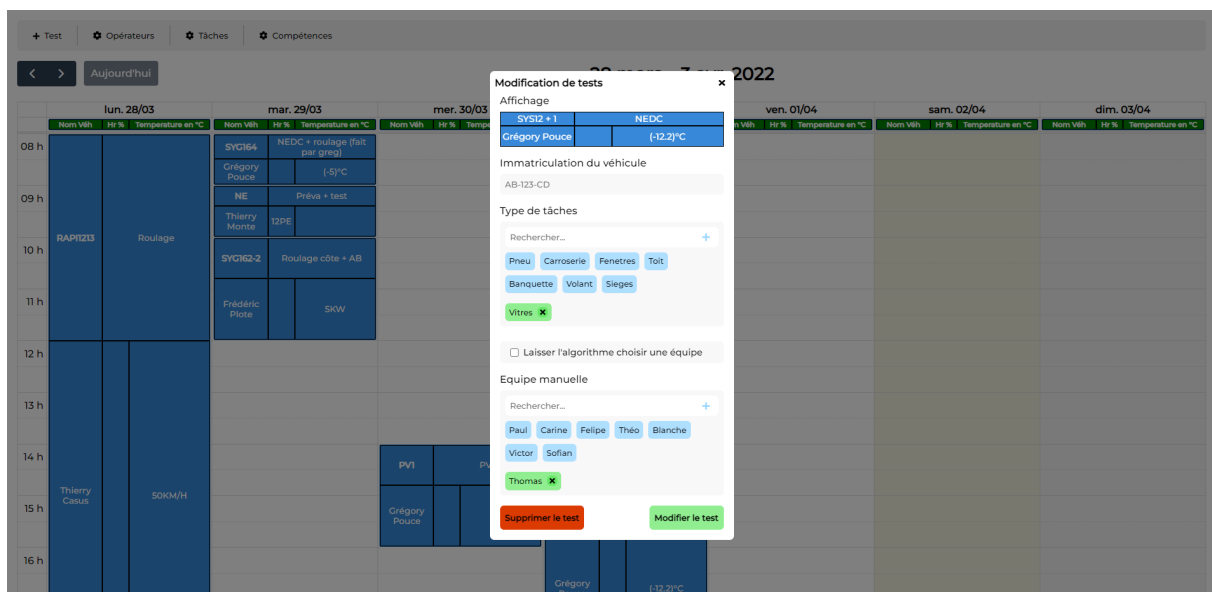


Figure 5.3 – Vue de modification d'un test

Depuis cette vue on peut aussi créer une tâche, et creer une équipe, respectant le levier n°2.

Enfin, pour présenter les derniers points voici la vue d'option des tâches, elle sert à créer et supprimer des tâches **Figure 5.4**.

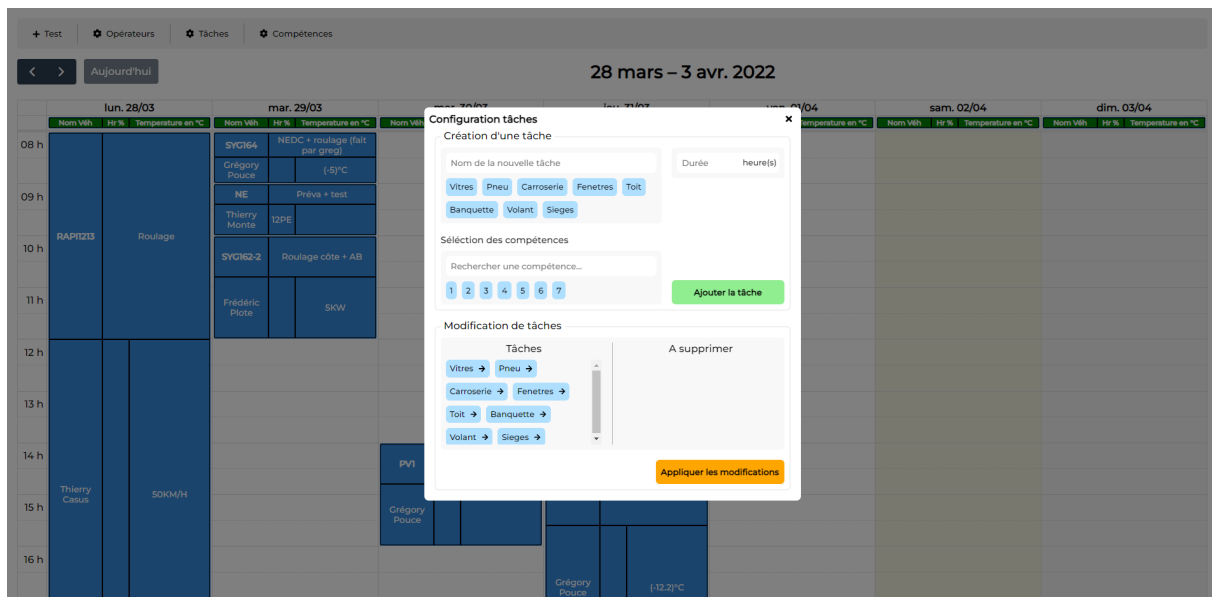


Figure 5.4 – Vue d'options des tâches

Cette vue permet aussi de présenter un autre aspect du levier n°1, la vue permet de supprimer plusieurs tâches à la fois, évitant de cliquer beaucoup de fois sur les boutons si l'on a de nombreuses tâches à supprimer.

Nous avons vu de quelles manières l'UX de l'application a été assurée.

Je ne m'attarderai pas sur le code de la partie front-end car peu intéressant à mon sens, je préfère plutôt le faire dans la partie solveur qui suit.

2 Partie solveur

Comme présenté dans la partie spécification (Annexe B), voici les différentes classes utilisées pour la partie solveur [Figure 5.5](#).

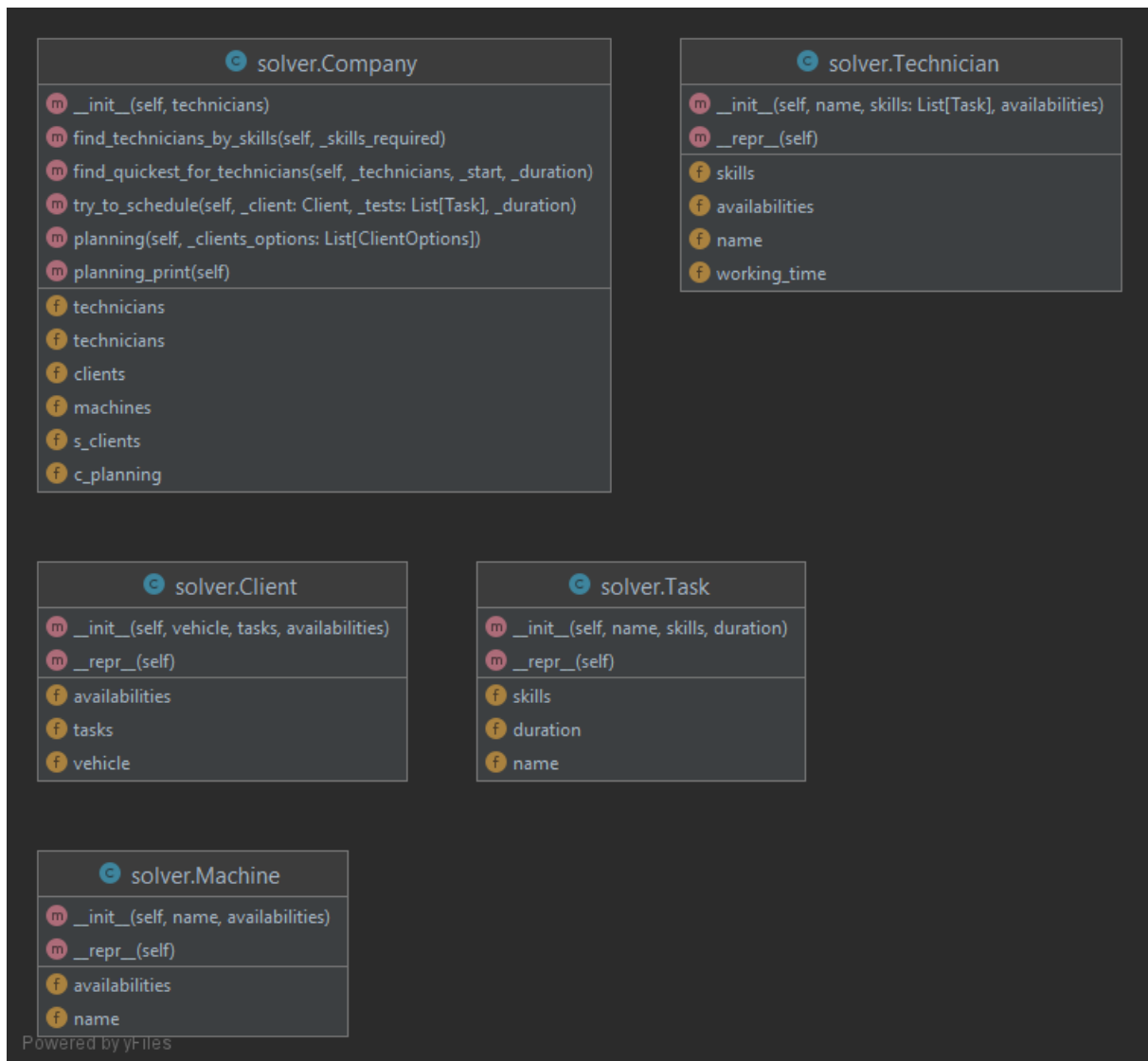


Figure 5.5 – Diagramme de classes du solveur

Il est important de comprendre que dans notre cas nous ne cherchons pas la solution optimale d'un problème de planification de tâches, nous nous contenterons de trouver une solution respectant au mieux les contraintes fixées. C'est pour cela que nous n'avons pas décidé de faire appel à la puissance d'outils commerciaux pour régler ce problème. La planification doit être réalisée en un temps raisonnable de maximum 10 secondes défini dans le cahier de spécifications.

Pour présenter le fonctionnement du solveur, voici le pseudo-code de son algorithme :

```

Trier les clients par ordre croissant de proximité de la date de rendu
Pour chaque client:
  On trouve les opérateurs ayant les compétences du test à effectuer
  On trouve les opérateurs pouvant finir le test au plus tôt
  Parmi les opérateurs restants on choisi celui qui a le moins travailler
  On attribue le test à l'opérateur

```

Figure 5.6 – Pseudo-code de l'algorithme du solveur

Ce pseudo-code est simplifié par rapport à la réalité de l'algorithme, mais permet néanmoins d'avoir une petite compréhension des mécaniques sous-jacentes.

Algorithm 1 Entry Point

```

clientOptions  $\leftarrow$  sort(listeDesClientOptions, minDueDate)
for clientOption  $\in$  clientOptions do
  client  $\leftarrow$  clientOption.client
  duration  $\leftarrow$   $\sum$  client.tasks.duration
  operator, dayNumber, period  $\leftarrow$  tryToSchedule(client, client.tasks, duration)
  operator.workingTime  $\leftarrow$  operator.workingTime + duration
  updatePlanning(operator, client, dayNumber, period, duration)
end for

```

Cet algorithme permet de récupérer un opérateur, un jour de la semaine, et une période de la journée pour attribuer un technicien à un test.

Algorithm 2 tryToSchedule(client, tasks, duration)

```

avail  $\leftarrow$  client.availabilities
operators  $\leftarrow$  findTechnicianBySkill(tasks.skills)
operators, dayNumber, period  $\leftarrow$  findQuickestForTechnicians(operators, avail, duration)
operator  $\leftarrow$  minWt(operators)
return  $\leftarrow$  operator, dayNumber, period

```

Cet algorithme renvoie le technicien, le jour de la semaine, la période de la journée respectant le mieux les contraintes.

Algorithm 3 findTechnicianBySkill(skills)

```

potentialOperators  $\leftarrow$  {}
for operator  $\in$  listeDesTechniciens do
  if (operator.skills  $\cap$  skills) = skills then
    potentialOperators  $\leftarrow$  operator
  end if
end for
return  $\leftarrow$  potentialOperators

```

Cet algorithme permet de trouver les techniciens ayant les compétences nécessaires pour effectuer une tâche.

L'algorithme de la partie findQuickestForTechnicians ne sera pas évoqué ici, car trop complexe.

Pour un exemple de résultat, voir la figure 6.3(Chapitre 6).

6

Bilan et conclusion

1 Bilan du semestre 9

Mes prédécesseurs ont déjà réalisé une bonne partie du projet même si beaucoup de choses sont à reprendre, certaines bases sont là.

J'ai commencé par réaliser un rapport d'avancement du projet (cf. Annexe C) où j'ai immédiatement compris que le projet ne pourrait pas être terminé au vu de l'énorme charge de travail qu'il restait encore. J'ai donc établi des priorités sur les tâches que je pouvais réaliser pour ce PRD. Je suis personnellement micro-entrepreneur dans le secteur de la création de sites et d'applications web, je pourrai donc plus aisément réaliser la partie Front-End. Le solveur et la partie Front-End seront donc mes objectifs pour le S10.

Voici le planning que je me suis fixé pour ce premier semestre, j'ai donc réalisé plus que ce que le planning initial :

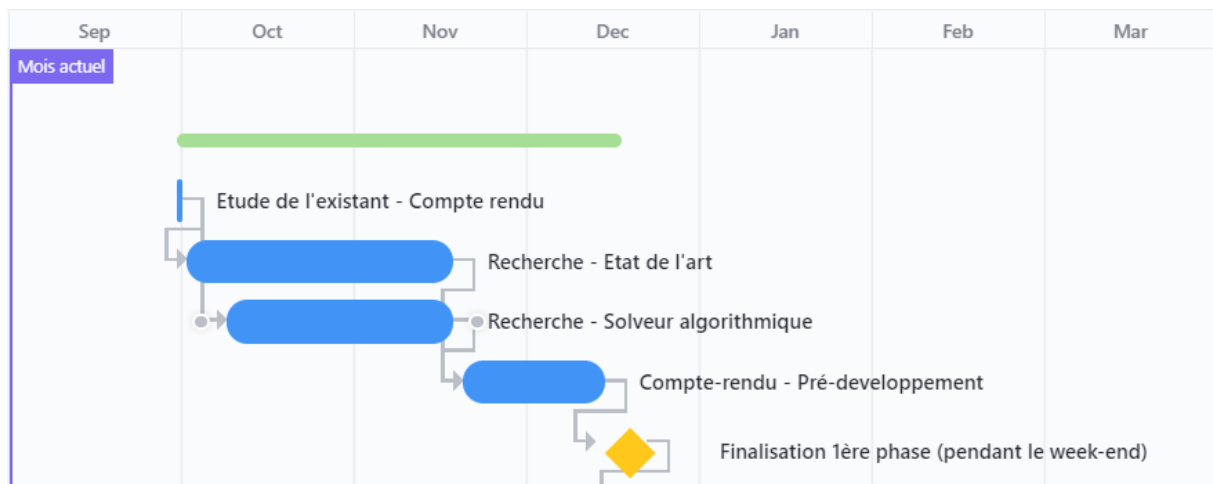


Figure 6.1 – Diagramme de Gantt réalisé au début du projet

Voici le planning que je me suis fixé à la fin du semestre 9, pour démarrer le semestre 10 :

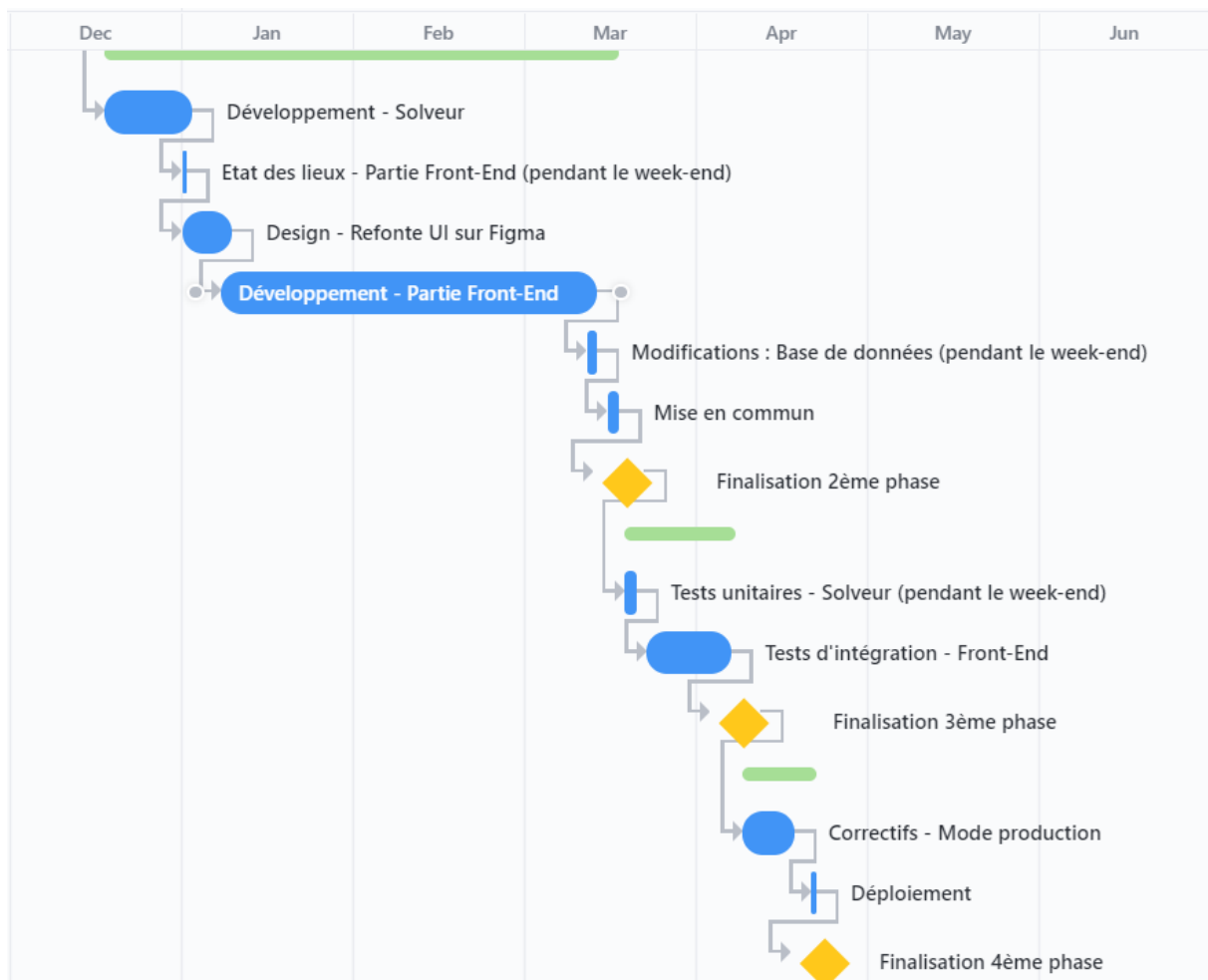


Figure 6.2 – Diagramme de Gantt réalisé en fin de S9

Le travail de mes prédécesseurs m'a permis de commencer en avance la phase de développement, j'ai ainsi développé un solveur partiel en Python ne prenant pas encore en compte toutes les contraintes. Ce solveur donne pour l'instant des prévisions sur une semaine.

Dont voici un extrait :

```

DAY0:
[[jey], [Ferari], 'complet']
[[joe], [Peugeot], 'pneu']
[[joe], [Peugeot], 'vitres']
[[joe], [Peugeot], 'complet']
[[jey], [Peugeot], 'pneu']
[[jey], [Peugeot], 'vitres']
[[jey], [Peugeot], 'complet']
[[jey], [Ferari], 'complet']

DAY1:
[[jey], [Ferari], 'complet']
[[joe], [Peugeot], 'pneu']
[[joe], [Peugeot], 'vitres']
[[joe], [Peugeot], 'complet']
[[joe], [Ferari], 'complet']
[[jey], [Ferari], 'complet']
[[joe], [Ferari], 'complet']
[[jey], [Ferari], 'complet']

DAY2:
[[joe], [Ferari], 'complet']
[[jey], [Ferari], 'complet']
[[joe], [Ferari], 'complet']
[[], [], 0]
[[], [], 0]
[[], [], 0]
[[], [], 0]
[[], [], 0]
[[], [], 0]

```

Figure 6.3 – *Extrait du résultat du solveur partiel*

Ici, chaque ligne représente un créneau de la journée

Je n'ai mis aucune information sur les données d'entrées volontairement, cette image n'est pas à titre démonstratif.

2 Bilan du semestre 10

Les objectifs les plus importants que je m'étais fixés ont été atteints, il faut aussi noter que d'autres n'ont pas pu être finis par manque de temps, voici le diagramme de Gantt final au S10 :

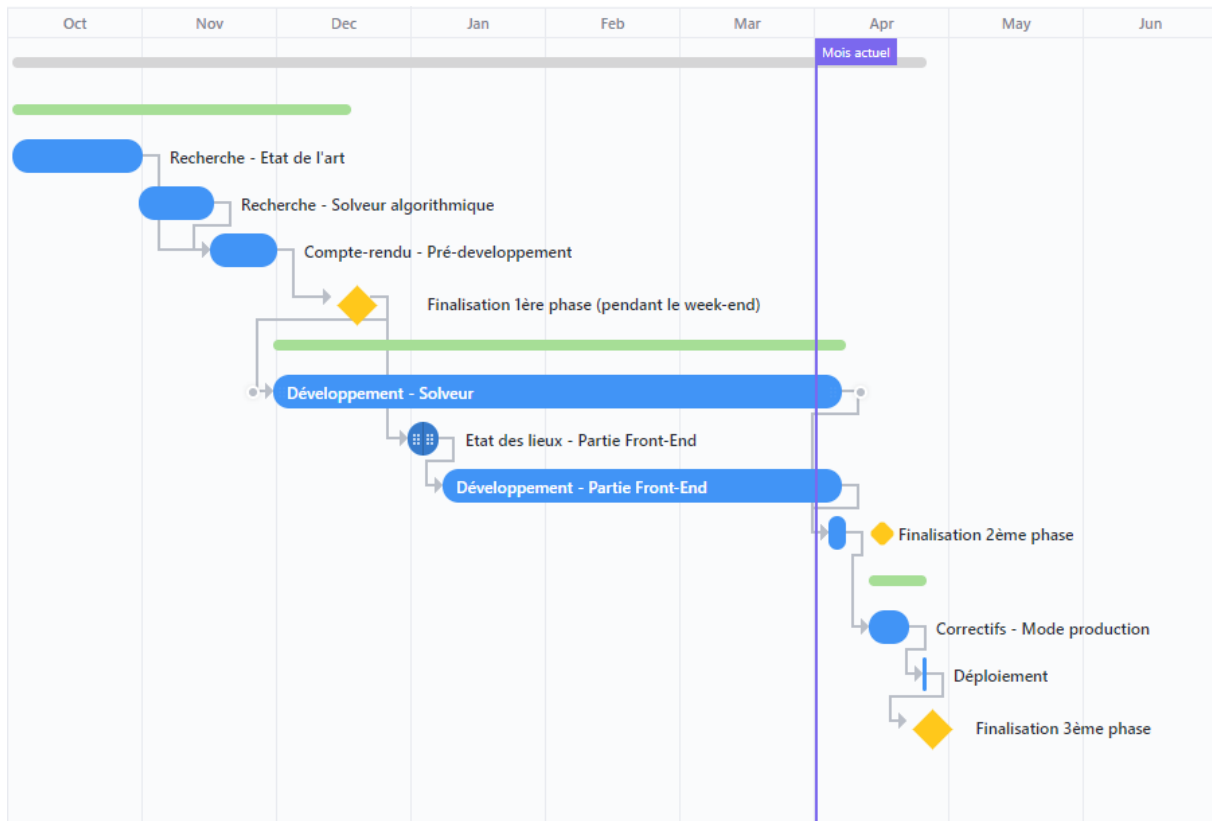


Figure 6.4 – Diagramme de Gantt réalisé en fin de S10

Pour résumer les objectifs atteints :

- solveur Python fonctionnel.
- partie Front-End réalisée mais pas connectée au Back-End.

3 Bilan auto-critique

Je suis content de travail effectué pour ce PRD, j'ai réussi à dépasser le cadre que nous nous étions fixé avec mon encadrant. Le projet n'est pas encore terminé, mais les prochaines équipes travaillant dessus pourront reprendre mon travail facilement avec les commentaires et la qualité mise en place.

Annexes

A

Planification, gestion de projet

1 Evolution du projet

Le diagramme de Gantt Initial pour la planification de ce projet

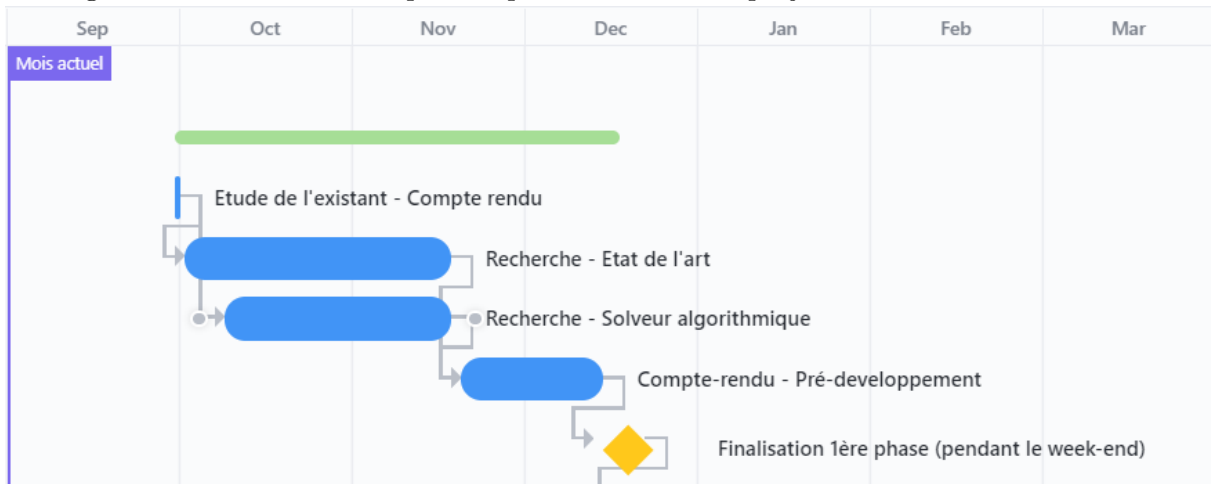


Figure A.1 – *Le diagramme de Gantt prévisionnel S9*

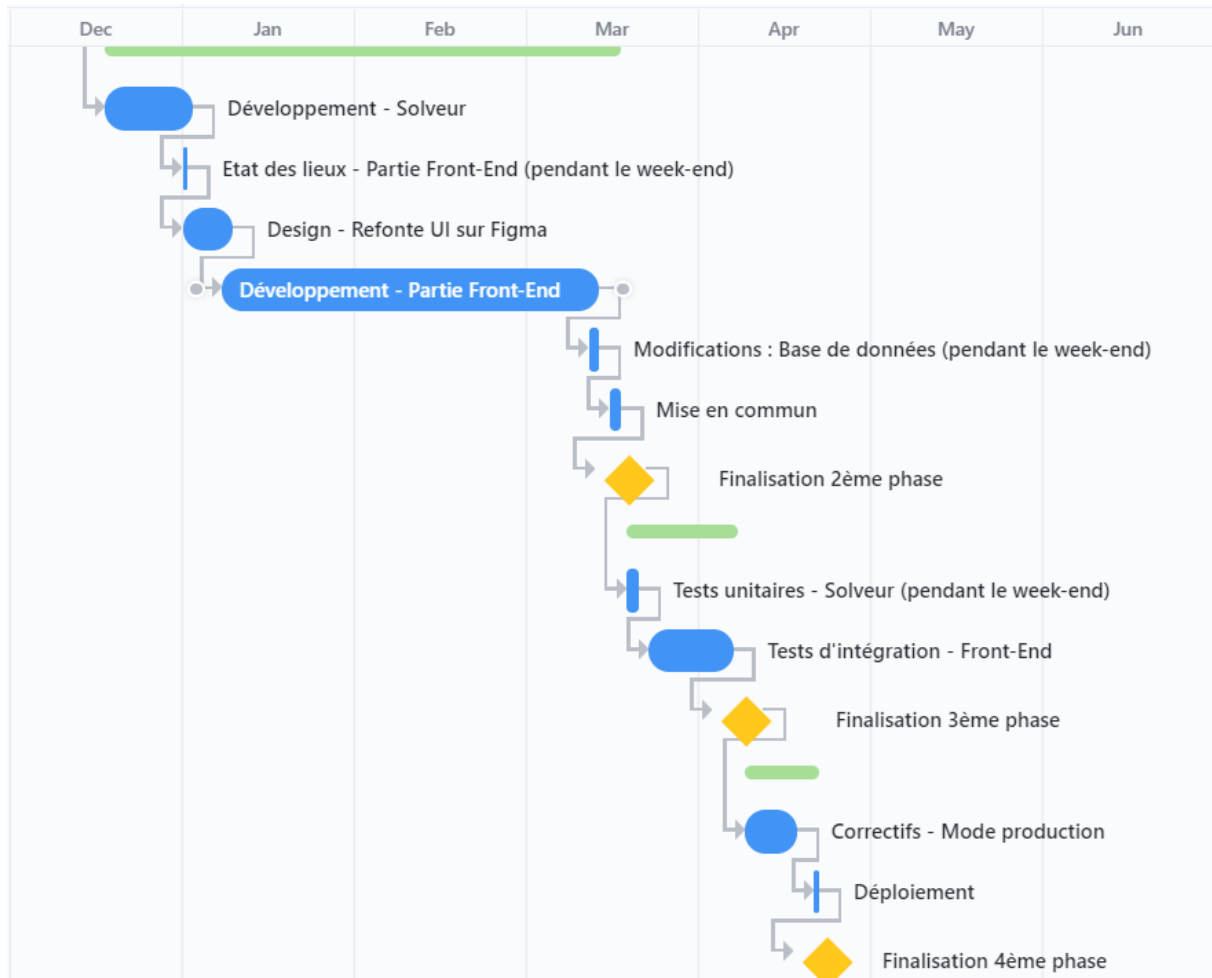


Figure A.2 – Le diagramme de Gantt prévisionnel S9-S10 mis à jour

B

Cahier de Spécifications

1 Solveur Algorithmique

Le solveur sera composé de 6 classes pour définir les objets liés au sujet. Autour s'articuleront des fonctions de tri, de calculs et d'affichage.

1.1 Les classes

Nous trouverons 6 classes pour représenter :

- Un opérateur
- Une machine
- Un client
- Une tâche
- Des options du client
- Une entreprise

Opérateur

Cette classe simule un opérateur, ses propriétés sont :

- Nom : Nom de l'opérateur
- Compétences : Tableaux de ses compétences
- Jours de travail : Ses horaires sur la semaine
- Temps de travail en cours : Nombre d'heures de travail pour une semaine

Machine

Cette classe simule une machine, ses propriétés sont :

- Nom : Nom de la machine
- Compétences : Tableaux des compétences qu'elle peut réaliser
- Départ et fin : Tableaux de tâches à exécuter en début ou fin d'une tâche
- Horaires : Horaires de fonctionnements

Client

Cette classe simule un client, ses propriétés sont :

- Nom : Nom du véhicule
- Tâches : Tableaux des tâches à réaliser
- Horaires : Disponibilités du client

Tâche

Cette classe simule une tâche, ses propriétés sont :

- Nom : Nom de la tâche
- Compétences : Tableaux des compétences nécessaires pour réaliser la tâche
- Durée : Durée de la tâche

Options

Cette classe simule des options spécifiques d'un client :

- Client : Client attaché
- Refaire : Nombre de fois que la tâche doit être effectuée

Entreprise

Cette classe simule l'entreprise :

- Opérateurs : Tableau d'opérateurs
- Clients : Tableau de clients
- Clients triés : Tableau de clients triés selon des KPI spécifiques
- Planning : Planning de sortie

1.2 Fonctions de tri

Ces fonctions servent à trier l'ordre d'attribution des tâches aux opérateurs, les opérateurs en fonction de le temps de travail actuel.

getMinWT

Entrée : Tableau d'opérateurs

Sortie : Tableau d'opérateurs triés par ordre croissant du temps de travail actuel

Préconditions : /

Postconditions : /

findOperatorBySkill

Entrées : Compétences nécessaires, Opérateurs

Sortie : Tableau d'opérateurs triés par ordre croissant de correspondance de compétences

Préconditions : /

Postconditions : /

1.3 Fonctions de calculs**calcDuration**

Entrée : Tableau de tâches

Sortie : Nombre d'heures de travail pour les réaliser

Préconditions : /

Postconditions : /

intersection

Entrées : Deux tableaux

Sortie : Intersection des deux tableaux

Préconditions : /

Postconditions : /

notIntersection

Entrées : Deux tableaux

Sortie : Elements n'étant pas dans l'intersection des deux tableaux

Préconditions : /

Postconditions : /

findQuickestForOperators

Entrées : Tableaux d'opérateurs pouvant réaliser la tâche, Horaire à partir de laquelle la

tâche peut commencer, durée de la tâche

Sortie : Tableau d'opérateurs potentiels, jour de la semaine ou la tâche peut commencer, heure du jour ou la tâche peut commencer

Préconditions : /

Postconditions : /

tryToSchedule

Entrées : Un client, tableau de tâches à réaliser, durée des tâches

Sortie : Opérateur qui va réaliser la tâche, jour de la semaine ou la tâche commencera, heure du jour ou la tâche commencera

Préconditions : /

Postconditions : /

planning

Entrées : Tableau d'option de clients

Sortie : Planning complet

Préconditions : /

Postconditions : /

1.4 Fonctions d'affichage

PlanningPrint

Entrée : Planning calculé

Sortie : Affichage dans la console

Préconditions : /

Postconditions : /

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

Il n'y a pas de contraintes de développement particulières, la partie **Front-End** sera développée avec le framework Vue.js version 3. Typescript sera préféré à Javascript du fait qu'il est plus facilement maintenable. En **Lint**, ESLint sera utilisé pour uniformiser l'écriture du code. Le code sera **Bundler** avec la librairie Vite choisie pour sa rapidité de bundle. Aucun framework graphique ne sera utilisé, les éléments seront désignés manuellement.

Le solveur n'a pas de contraintes spécifiques de développement.

2.2 Contraintes de fonctionnement et d'exploitation

Dans cette section sont décrites les dispositions qu'il est nécessaire de prendre en compte pour les différentes conditions de fonctionnement du système.

2.2.1 Performances

Un temps de réponse inférieur à 10s peut être envisageable pour ce genre de projet, les contraintes de performances ne sont pas strictes, mais elles doivent être raisonnables.

De ce fait le temps de réponse de solveur se doit lui aussi de respecter une limite de 10s. Sachant que le calcul du solveur sera la partie de l'application qui sera la plus lente.

2.2.2 Contrôlabilité

Il est nécessaire de simplifier l'application pour le client en lui permettant de faire des actions de manière intuitive sans grande connaissance de l'application.



Rapport d'avancement

Ce rapport présente l'avancement général du travail effectué par mes prédécesseurs.

Rapport d'étude de l'outils existant

1. Base de données

CRITERES D'AVANCEMENT

OPERABILITE	FONCTIONNELLE
MAINTENABILITE	ASSUREE
SCALABILITE	ASSUREE
COMPLEXITE	MOYENNE

Notes : Vider la base de données : ne pas rentrer de données, mots de passes en clairs : très mauvais.

2. Back-End

CRITERES D'AVANCEMENT

OPERABILITE	FONCTIONNELLE
MAINTENABILITE	ASSUREE
SCALABILITE	ASSUREE
COMPLEXITE	HAUTE

Notes : Nécessité de tester avec un front-end valide

3. Front-End

CRITERES D'AVANCEMENT

OPERABILITE	NON-FONCTIONNELLE
MAINTENABILITE	FAIBLE
SCALABILITE	TRES FAIBLE
COMPLEXITE	MOYENNE

Notes :

- Nécessité de refaire ou de faire une refonte de l'UI/UX : expérience utilisateur mauvaise (pour ne pas dire inutilisable).
- Nécessité d'améliorer l'opérabilité : Ajout de Tests : Non fonctionnels
- Nécessité d'améliorer la scalabilité : Importations Excel, Vuex.
- Nécessité d'améliorer la maintenabilité : Langages typés, tests

Conclusions :

La charge de travail sur la partie front-end est très élevée (à refaire), nécessiterait sûrement un projet à part entière. Les parties restantes sont fonctionnelles, il n'y a rien à refaire sauf quelques petites modifications (cf. Notes).

Figure C.1 – Rapport d'avancement

D

Document d'installation

Le projet se télécharge depuis git directement dans l'IDE, ou bien sous la forme d'une archive à extraire dans le répertoire de travail choisi.

La partie Front-End utilise le gestionnaire de librairies NPM, pour installer les librairies il faudra ouvrir un terminal dans le repertoire du projet et lancer la commande : `npm install`

Pour la partie solveur il faudra ouvrir un terminal dans le repertoire du projet et lancer la commande : `pip install numpy`

E

Document d'utilisation

Paramétrer l'environnement du problème que l'on souhaite dans le code, puis lancer le solveur avec la commande : `python ./solver.py`

Celui-ci renvoi le planning dans stdout.

Pour la partie Front-End, lancer le solveur avec la commande : `npm run dev`

Pour accéder à l'interface se rendre à l'adresse : `http://localhost:3000/` avec un navigateur.

Depuis cette adresse, se rendre directement à l'url : `/home` pour accéder à l'emploi du temps.

F

Cahier de test

Pour lancer les tests de la partie solveur, ouvrir un terminal dans le dossier du projet, lancer la commande : `python ./tests.py`

Résultats :

Unittests for tests.SolverTests: 7 total, 7 passed			15 ms
			Collapse Expand
tests			15 ms
SolverTests			15 ms
test_calc_duration_ok (Test that the sum of a list of task's duration is right)	passed	0 ms	
test_company_find_quickest_for_technician (Test that the function returns the rights values (quickest technician, quickest day number, correct duration))	passed	0 ms	
test_company_find_technicians_by_skills (Test that the function returns the rights values (technician with according skill sets))	passed	0 ms	
test_company_try_to_schedule (Test that the function returns the rights values (quickest technician, quickest day number, correct duration))	passed	0 ms	
test_get_min_wt_ok (Test that the function returns the technician with the lowest working_time)	passed	0 ms	
test_intersection_ok (Test that the function returns the real intersection of arrays)	passed	15 ms	
test_not_intersection_ok (Test that the function returns the elements outside of the intersection of the arrays)	passed	0 ms	

Figure F.1 – Résultats des tests



Bibliographie

- [1] Cheikh DHIB. « Etude et résolution de problèmes d'ordonnancement de projets multi-compétences : intégration à un ERP libre. (Study and resolution of multi-skill project scheduling problems : integration on open-soure ERP) ». Thèse de doct. François Rabelais University, Tours, France, 2013. URL : <https://tel.archives-ouvertes.fr/tel-01367949>.
- [2] Arthur PELLOUX-PRAYER. « Solution web service pour la conception d'un planning collaboratif ». Projet Recherche & Développement. Tours, France : Ecole Polytechnique de l'Université de Tours, 2020.

H

Glossaire

Back-End Etage d'une application web effectuant les opérations demandées par l'utilisateur. [1](#), [3](#)

Bundler Librairie chargée de 'compiler' le projet. [27](#)

Framework Ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns. [1](#)

Front-End Etage d'une application web servant d'interface au Back-End, elle lui transmet les opérations demandées par l'utilisateur . [1](#), [3](#), [27](#)

Lint Librairie chargée d'uniformiser les styles de codes afin de rendre l'application plus facilement maintenable. [27](#)

I

Acronymes

CNSKILL Closest Number of Skill. 9

EDD Earliest Due Date. 9

KPI Key Performance Indicator. 6

LB Load Balancing. 9

MVC Modele, Vue, Controlleur. 11

PRD Projet de Recherche et Développement. 1

UX User eXperience. 13, 15

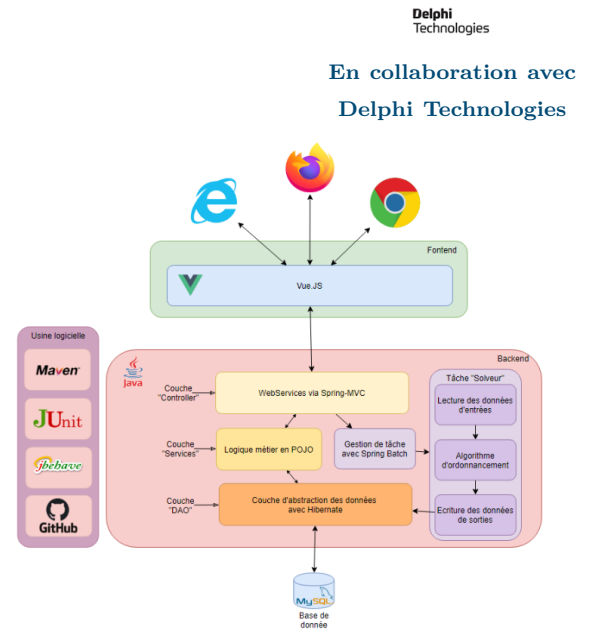
Solution web service pour la conception d'un planning collaboratif

Nicolas MAGNE

Encadrement : Ameur SOUKHAL

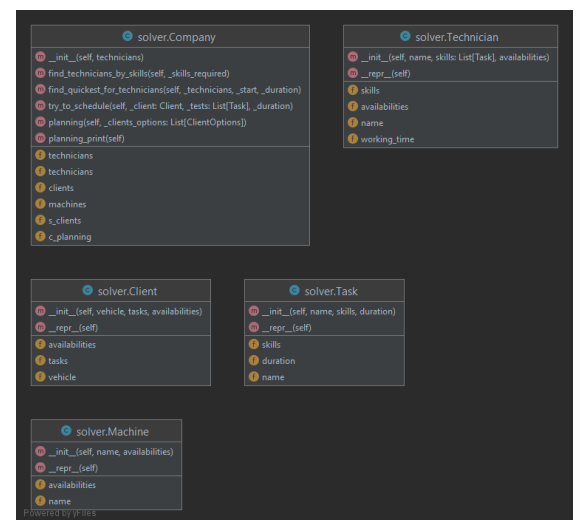
Objectifs

- Définition de nouveaux indicateurs de performances servant de contraintes à un solveur algorithmique.
- Développement d'un solveur algorithmique portant sur la question d'optimisation d'ordonnancement de projets multi-compétences.
- Développement d'une interface web professionnelle.



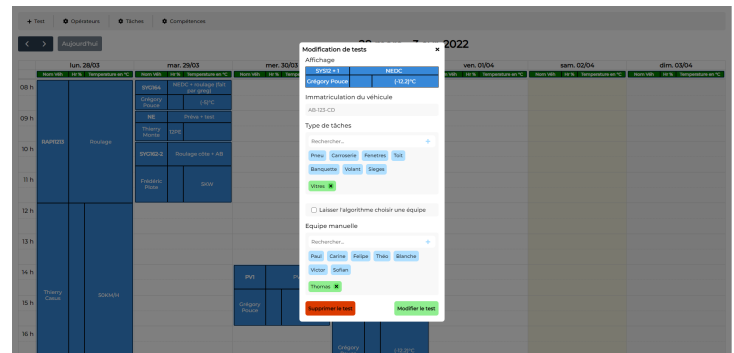
Mise en œuvre

1. Création d'un KPI fusionnant (Load Balancing, Earliest Due Date, Closest Number of Skill)
2. Solveur algorithmique en Python.
3. Interface web en Vue.js



Résultats attendus

1. Solveur algorithmique renvoyant une solution possible mais pas optimale pour le problème.
2. Interface web moderne et professionnelle, facile à prendre en main.



Solution web service pour la conception d'un planning collaboratif

Nicolas MAGNE

Encadrement : Ameer SOUKHAL

Delphi Technologies

En collaboration avec Delphi Technologies

Objectifs

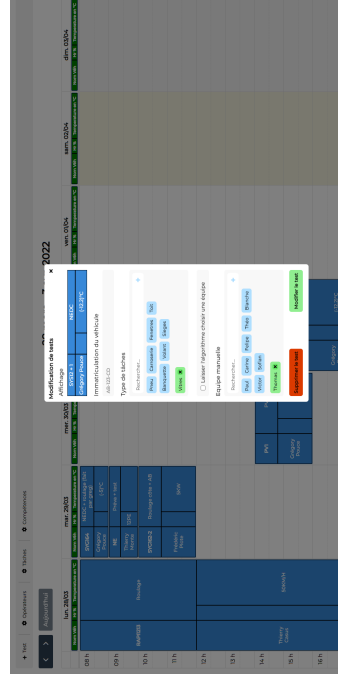
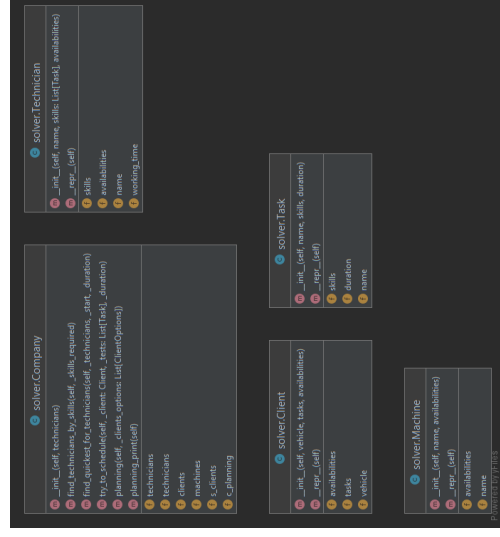
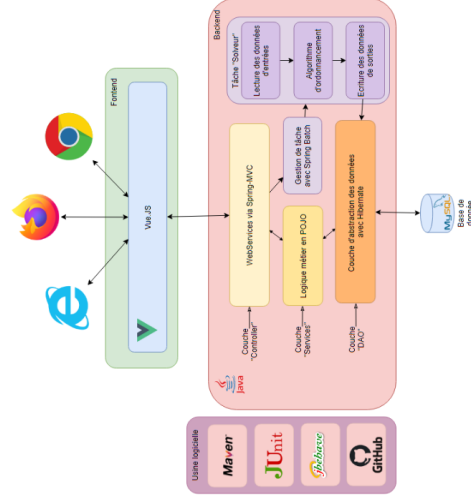
- Définition de nouveaux indicateurs de performances servant de contraintes à un solveur algorithmique.
- Développement d'un solveur algorithmique portant sur la question d'optimisation d'ordonnement de projets multi-compétences.
- Développement d'une interface web professionnelle.

Mise en œuvre

1. Création d'un KPI fusionnant (Load Balancing, Earliest Due Date, Closest Number of Skill)
2. Solveur algorithmique en Python.
3. Interface web en Vue.js

Résultats attendus

1. Solveur algorithmique renvoyant une solution possible mais pas optimale pour le problème.
2. Interface web moderne et professionnelle, facile à prendre en main.



Solution web service pour la conception d'un planning collaboratif

Résumé

Voici le résumé de ce PRD.

Mots-clés

motcle1, motcle2, etc.

Abstract

Here is the abstract of this project.

Keywords

word1, word2, etc.

Entreprise

Delphi Technologies

Delphi
Technologies

Tuteur entreprise

Étudiant

Nicolas MAGNE (DI5)

Tuteur académique

Ameur SOUKHAL