

Rapport du projet

***Projet : Conception d'un outil d'expérimentation
de scénarios et de génération d'emploi du temps***

Arbaut Jean-Baptiste

Deschamps Kylian

Duez-Faurie Valentine

Eramil Kadir

Pilloud Aubry

Tropel Célia


Université Grenoble Alpes

Rapport du projet

Conception d'un outil d'expérimentation de scénarios et de génération d'emploi du temps

Les informations d'identification du document :

Les éléments de vérification du document :

Référence document :	du	Rapport du projet
Version document :	du	1.1
Date du document :	17/06/2025	
Auteur(s) :	Arbaut Jean-Baptiste Deschamps Kylian Duez-Faurie Valentine Eramil Kadir Pilloud Aubry Tropel Célia	
Validé par : Aurélie Landry		
Validé le : 17/06/25		
Soumis le : 15/06/2025		
Type de diffusion : Document électronique (.pdf)		
Confidentialité : /		
<i>Les éléments d'authentification :</i>		
Maître d'ouvrage:	Aurélie Landry	
Date / Signature :	17/06/25	
Chef de projet :	/	
Date / Signature :	/	

Sommaire

Sommaire.....	3
Remerciements.....	4
1. Introduction.....	4
1.1 Objectifs et méthodes.....	4
2. Guide de lecture.....	5
3. Contexte du projet.....	6
3.1 Origine du projet.....	6
3.2 Évolution des objectifs.....	6
4. Organisation du projet.....	6
5. Technologies utilisées.....	7
6. Résumé des réalisations du projet.....	7
6.1 Réalisations côté interface.....	7
6.2 Réalisations côté solveur.....	7
7. Difficultés rencontrées.....	8
7.1 Problèmes liés à l'interface.....	8
7.2 Complexités du solveur.....	8
7.3 Problèmes d'intégration interface / solveur.....	9
7.4 Contraintes de gestion de projet.....	9
7.5 Limites de Monoposte.....	9
7.6 Adaptation du cahier des charges.....	9
8. Perspectives d'améliorations.....	10
8.1 Améliorations du solveur.....	10
8.2 Améliorations de l'interface.....	10
8.3 Améliorations générales.....	10
8.4 Améliorations organisationnelles.....	10
8.5 Perspectives fonctionnelles.....	11
9. Dysfonctionnement du solver.....	11
10. Glossaire.....	11
11. Références.....	12
12. Index.....	13
13. Annexes.....	14

Remerciements

Nous tenons à remercier Mme Landry, maître d'ouvrage du projet, qui a également assuré un rôle de référente tout au long de l'année. Son accompagnement, ses retours réguliers et sa disponibilité ont été essentiels à l'orientation et à l'avancement de notre travail.

Nos remerciements vont également à M. Pellier, pour ses conseils techniques tout au long du premier semestre et au début du second. Il nous a apporté un cadre méthodologique rigoureux et des recommandations précieuses. Plus largement, nous remercions l'ensemble du jury de la soutenance du premier semestre pour leurs retours sur le projet et la présentation.

Enfin, nous souhaitons remercier M. Laffond, principal d'un collège, pour le temps qu'il nous a consacré afin de nous expliquer en détail les pratiques actuelles de création d'emplois du temps dans son établissement. Il nous a également transmis des jeux de données réels (anonymisés), qui ont été utiles pour évaluer la volumétrie d'informations ainsi que pour nos tests.

1. Introduction

Ce document constitue le rapport final du projet de TER mené par notre équipe de six étudiants en Master 1 MIASHS, parcours Informatique et Cognition. Il présente l'ensemble du travail réalisé autour du développement d'une application de génération automatique d'emplois du temps pour les collèges.

L'objectif de ce rapport est de documenter l'ensemble du projet, depuis le contexte initial jusqu'aux résultats obtenus, en passant par l'organisation du travail, les difficultés rencontrées et les perspectives d'amélioration. Le lecteur y trouvera une description de l'application développée, composée d'une interface web et d'un moteur de résolution de contraintes (solveur).

1.1 Objectifs et méthodes

L'application développée s'inscrit dans le cadre d'un projet de TER (Travail d'Étude et de Recherche) et vise à proposer une solution complète et accessible pour la génération automatique d'emplois du temps dans les établissements scolaires de type collège. Cette application a pour objectif de réduire significativement le temps et la complexité liés à la création manuelle des plannings hebdomadaires, tout en garantissant la prise en compte de l'ensemble des contraintes pédagogiques, matérielles et humaines propres à chaque établissement.

Le projet repose sur deux composantes principales :

- Une interface web interactive, conçue avec le framework Dash (Python), permettant à l'utilisateur de renseigner ou importer les données nécessaires : enseignants, classes, disciplines, salles, contraintes horaires, options pédagogiques... Cette interface se veut claire, et accessible à des utilisateurs sans compétences en programmation. Elle permet également de visualiser, modifier et exporter les résultats.
- Un moteur de résolution de contraintes, basé sur la bibliothèque OR-Tools développée par Google, appelé "solveur", qui modélise le problème de l'emploi du temps sous forme d'un problème d'optimisation à contraintes. Ce solveur prend en compte les règles définies (volumes horaires, incompatibilités, préférences, simultanés, etc.) et produit des plannings faisables et équilibrés, en minimisant les conflits.

Le développement a suivi une approche itérative en plusieurs étapes :

1. Recueil des besoins auprès d'exemples réels d'emplois du temps scolaires et définition des types de contraintes à prendre en charge.
2. Modélisation du problème et du plan de développement.
3. Implémentation du solveur avec OR-Tools, incluant la gestion des sous-groupes, des salles, des volumes horaires par matière, et des conflits potentiels. Conception en parallèle de l'interface Dash en modules fonctionnels : saisie des données, gestion des contraintes, visualisation des résultats et export. Avec des fonctionnalités telles que le système de sauvegarde automatique.
4. Validation continue par des jeux de données réalistes et ajustements selon la faisabilité et la lisibilité des emplois du temps générés.
5. Création des tests et de la documentation associée.
6. Documentation : les commentaires du code source ainsi que la documentation interne ont été réalisés tout le long du projet. Le rapport final et les guides ont été rédigés en fin de projet.

L'ensemble du système est conçu pour fonctionner localement, sans dépendre d'un serveur distant, et pour pouvoir être adapté selon les spécificités des établissements. Le projet met l'accent sur la modularité, la fiabilité des résultats, et l'accessibilité pour des non-informaticiens. L'un des objectifs était de fournir un export compatible avec Monoposte d'Index Education, ce qui n'a pas pu être effectué.

2. Guide de lecture

Ce rapport s'adresse à tout lecteur souhaitant comprendre les objectifs, le déroulement et les résultats du projet. L'ensemble des sections est conçu pour être lu dans sa globalité afin d'offrir une vision complète du travail réalisé.

3. Contexte du projet

Ce projet a été mené par six étudiants de Master 1 MIASSH, parcours Informatique et Cognition, dans le cadre d'un Travail d'Étude et de Recherche au laboratoire d'informatique de Grenoble (LIG, bâtiment IMAG). Le projet est sous la supervision de Mme Landry, maître d'ouvrage, qui nous a fait part d'un besoin concret concernant la génération d'emplois du temps dans un établissement scolaire.

3.1 Origine du projet

L'idée initiale du projet est née d'une situation réelle : dans un collège, les emplois du temps comportaient de nombreuses heures de permanences pour les élèves comme pour les enseignants. Il a été supposé que les emplois du temps étaient construits manuellement, ou du moins, avec des outils peu adaptés.

En approfondissant, nous avons découvert que l'établissement disposait bien d'un outil performant de génération automatique d'emplois du temps. La personne chargée de la création des emplois du temps étant partie, c'est son successeur qui a présenté le fonctionnement de l'outil et qui a expliqué qu'il parvenait à des résultats satisfaisant sans exploiter pleinement ses fonctionnalités. Peu de limites ont été exposées par la personne, on peut retenir uniquement le temps de calcul relativement long, impossibilité d'intégrer certaines contraintes récentes (comme le poids des cartables).

3.2 Évolution des objectifs

Face à ces constats, l'objectif du projet a évolué. Plutôt que de simplement produire un nouvel emploi du temps, nous avons choisi de développer un outil de simulation flexible, capable d'intégrer un grand nombre de contraintes, y compris des contraintes nouvelles souvent négligées dans les outils classiques (comme la limitation du poids du matériel scolaire). L'enjeu est donc de créer une solution :

- Flexible, permettant d'intégrer des contraintes complexes.
- Explorable, afin d'offrir à l'utilisateur une explication claire en cas de non-respect d'une contrainte.
- Ergonomique, avec une interface simplifiée.

4. Organisation du projet

Durant le premier semestre nous avons utilisé l'outil Trello pour la gestion de projet, combiné à un diagramme de Gantt. Pour le second semestre, l'équipe projet s'est scindée en deux groupes distincts : deux personnes se sont chargées de l'interface utilisateur, tandis que les quatre autres ont travaillé sur le solveur. Le travail a été réparti de manière structurée : l'équipe interface s'est organisée par pages (saisie des données, contraintes, contraintes optionnelles, visualisation et export), et l'équipe solveur par contraintes à implémenter (disponibilités, volume horaire, poids du cartable...). Des échanges réguliers entre les deux

équipes ont été instaurés afin d'aligner les fonctionnalités de l'interface sur ce qui était réalisable techniquement dans le solveur et de définir les formats d'entrée et de sortie. En annexes, se trouve l'échéancier ([diagramme de Gantt](#)) du second semestre du projet.

5. Technologies utilisées

Le projet s'appuie principalement sur deux composantes : une interface web développée avec Dash (framework Python) et un solveur de contraintes construit avec OR-Tools, la bibliothèque d'optimisation de Google (Open-source). Le langage Python a été utilisé pour l'ensemble du développement, aussi bien pour l'interface que pour le solveur. Les données d'entrée et de sortie sont structurées en JSON, facilitant les échanges entre modules. Pour l'export des emplois du temps, une génération automatique de fichiers PDF a été mise en place. Le code du projet est partagé via GitHub.

6. Résumé des réalisations du projet

6.1 Réalisations côté interface

L'interface permet à l'utilisateur de saisir l'ensemble des données nécessaires à la génération d'un emploi du temps complet : enseignants, classes, matières, langues, salles, horaires, et volumes horaires issus du programme national. Elle prend également en charge la définition des contraintes propres à l'établissement, qu'elles concernent les indisponibilités des professeurs, des élèves et des salles ou le planning global, par exemple une obligation de cours à un moment clé. Les indisponibilités partielles peuvent être précisées, tout comme les contraintes dites optionnelles (poids du matériel, permanences, restauration scolaire), que l'utilisateur doit hiérarchiser selon leur importance. Une fois la saisie terminée, l'interface structure les données pour les rendre compatibles avec le solveur et appelle celui-ci. À l'issue du calcul, elle affiche les emplois du temps générés, permet leurs modifications, propose des statistiques associées comme le pourcentage de respect des contraintes, donne le détail des violations de contraintes et permet d'exporter les résultats au format PDF.

6.2 Réalisations côté solveur

Le solveur permet de générer des emplois du temps tout en respectant un grand nombre de contraintes. Il garantit notamment le respect des volumes horaires par matière et par professeur, la gestion des indisponibilités des professeurs et des salles, ainsi que les préférences horaires ou de salles exprimées. Il affecte correctement les salles spécialisées (informatique, laboratoire, etc.) aux matières concernées, et gère la répartition des cours entre un ou plusieurs enseignants. Les sous-groupes de langues et d'options sont également pris en compte, tout comme la surcharge de la permanence et de la cantine, qui peut déclencher une répartition automatique des élèves selon un pattern défini dans le fichier JSON.

Le solveur veille à ce qu'aucun professeur ne soit assigné à deux salles en même temps, et qu'un même cours ne soit pas présent dans deux classes à la fois, y compris entre niveaux différents. Il génère les emplois du temps pour les classes, mais aussi pour les professeurs et les salles, et produit un rapport de contraintes précisant celles qui sont respectées ou non.

Il est capable de traiter des contraintes complexes telles que : enchaînement obligatoire ou interdit entre deux matières, placement synchronisé d'un cours pour tout un niveau, limitation du poids des cartables avec tolérance de 5 %, gestion des journées sans après-midi, placement forcé de certains cours sur un créneau donné, ou nombre de présences imposées pour une matière sur une journée ou une demi-journée. Il gère également les semaines A/B, afin de respecter les volumes horaires hebdomadaires à demi-heure (ex : 2.5 h / semaine soit 5 h sur deux semaines). Il cherche à minimiser les heures de permanence dans la mesure du possible. Enfin, le système peut lancer plusieurs résolutions successives et sélectionner automatiquement celle qui respecte le plus grand nombre de contraintes, tout en estimant les temps d'exécution pour aider à la prise de décision.

7. Difficultés rencontrées

Le développement de ce projet a soulevé plusieurs types de difficultés, tant sur le plan technique que sur celui de la gestion de projet. Ci-dessous, sont présentés les principaux défis identifiés au cours de la réalisation.

7.1 Problèmes liés à l'interface

Le développement de l'interface avec Dash a présenté plusieurs défis techniques et ergonomiques. L'un des problèmes récurrents concernait les rafraîchissements déclenchés trop rapidement, qui provoquaient des erreurs liées à des variables non instanciées ou des objets encore absents de la mémoire. Pour y remédier, nous avons dû prévoir des initialisations de valeurs d'objets.

Un autre point de friction a été l'utilisation des Dash DataTables, pratiques pour structurer les données, mais très rigides en matière de mise en forme. Leur configuration s'est révélée peu adaptée pour des saisies ou pour simuler un affichage de type planning.

L'uniformisation visuelle entre les différentes pages a également posé problème. Certaines bibliothèques de composants ou extensions visuelles utilisées n'étaient pas compatibles entre elles, ce qui a nécessité des ajustements fréquents.

7.2 Complexités du solveur

La modélisation de certaines contraintes dans le solveur s'est révélée particulièrement complexe. C'est notamment le cas de la gestion des groupes de langues vivantes, dans lesquels les élèves sont répartis selon la langue choisie, parfois de manière transversale à plusieurs classes (par exemple : le groupe "5ESP1" regroupe des élèves des 5e1, 5e2 et 5e3). Il a donc fallu mettre en place une logique de regroupement cohérente, tout en maintenant le respect des autres contraintes, en particulier le volume horaire hebdomadaire.

Un autre défi majeur a concerné l’alternance des semaines A et B, il était nécessaire de respecter les volumes horaires en limitant les différences entre les semaines.

7.3 Problèmes d’intégration interface / solveur

Au début du projet, nous avons envisagé de définir une structure de données commune entre l’interface et le solveur, dans le but de faciliter les échanges. Cette approche s’est révélée inadaptée en raison du développement parallèle des deux modules, qui évoluaient à des rythmes différents. Finalement, nous avons opté pour une solution consistant à transformer les données de l’interface avant leur transmission au solveur.

7.4 Contraintes de gestion de projet

La répartition initiale des tâches s’est avérée sous-optimale : deux personnes pour l’interface, quatre pour le solveur. Ce choix, motivé à la fois par les préférences individuelles et par la perception du solveur comme une “boîte noire” difficile à découper, a compliqué l’organisation. Le découpage du solveur par contraintes a permis une certaine parallélisation, mais le poids du développement de l’interface avait été sous-estimé. De plus, l’absence de modularité dans certaines parties de l’interface ne nous a pas permis d’ajouter les options progressivement, ce qui a empêché la réalisation de tests utilisateurs tant que l’interface n’était pas suffisamment avancée, malgré un solveur déjà opérationnel.

7.5 Limites de Monoposte

Nous pensions initialement pouvoir échanger des données avec le logiciel Monoposte (Index Éducation), utilisé dans certains établissements. Cependant, ce dernier ne propose ni documentation publique ni format ouvert, ce qui a rendu toute tentative d’interopérabilité impossible. Nous avons donc dû concevoir notre propre système d’affichage et d’export des emplois du temps, afin de rendre l’application pleinement exploitable.

7.6 Adaptation du cahier des charges

Le cahier des charges initial définissait une interface très structurée, avec des paramètres censés être directement pris en compte par le solveur. Au fil du développement, nous avons dû simplifier certaines parties de l’interface et en ajouter d’autres pour répondre à l’évolution du solveur. Cette évolution itérative a nécessité une révision fréquente de la structure initiale, pour maintenir la cohérence entre ce que l’interface permettait de saisir et ce que le solveur pouvait effectivement exploiter.

8. Perspectives d'améliorations

8.1 Améliorations du solveur

Concernant le solveur, une amélioration nécessaire serait la gestion des cours de 30 minutes ou d'1h30, mais plus généralement, il serait pertinent de permettre à l'utilisateur de modifier la durée des créneaux de cours et les horaires d'ouverture de l'établissement pour les jours particuliers (généralement le mercredi). Cela correspond mieux à la réalité du métier.

Un autre ajout intéressant aurait été la possibilité de reprendre un emploi du temps incomplet afin de générer une nouvelle solution à partir de celui-ci.

Un dernier ajout du côté du solveur aurait pu être la prise en compte des emplois du temps des surveillants scolaires. En effet, ces personnels constituent une ressource essentielle au bon fonctionnement de l'établissement. Les temps de récréations, d'étude et de restauration ne dépendent pas uniquement de la capacité et de la disponibilité des salles : il est nécessaire d'avoir assez de personnel pour assurer la surveillance. Cela peut impliquer la gestion simultanée de plusieurs espaces d'études (foyer, étude, amphithéâtre), de la cour de récréation, du réfectoire, du portail pour les entrées et sorties, ainsi que de la vie scolaire.

8.2 Améliorations de l'interface

Pour l'interface, il aurait été utile d'intégrer des options de gestion de l'historique des générations d'emplois du temps, ainsi qu'un système permettant de vérifier que les modifications apportées côté interface respectent bien les contraintes définies à l'origine, ou, à défaut, d'alerter l'utilisateur afin de l'aider dans sa prise de décision.

8.3 Améliorations générales

Plus globalement, il serait possible d'optimiser davantage le temps de calcul afin d'améliorer la fluidité de l'application, ainsi que d'affiner la qualité du code et le rendu de l'interface. La documentation, elle aussi, peut toujours être perfectionnée.

8.4 Améliorations organisationnelles

D'un point de vue organisationnel, nous aurions souhaité faire réaliser des tests utilisateurs par différentes secrétaires de scolarité afin d'obtenir des retours critiques, mais cela n'a pas été possible. Le projet étant ambitieux, notre application n'a atteint un niveau de maturité suffisant que tardivement, ce qui ne nous a pas permis de solliciter le principal du collège pour proposer une vraie alternative à son logiciel de création d'emplois du temps.

8.5 Perspectives fonctionnelles

Enfin, une consécration aurait été de permettre à l'utilisateur de définir de nouvelles contraintes de façon interactive, aussi bien dans l'interface que pour le solveur.

9. Dysfonctionnement du solver

La fonction `transformer_interface_vers_config` retourne un fichier .JSON de résultat qui est correcte mais le solver est incapable de bien l'interpréter. Voici les problèmes que nous avons identifiés qui rendent le solver inutilisable :

- On ne peut pas mettre plus de 2 classes par niveau.
- Des changements minimes de volume horaire des matières (de l'ordre de 0,5h) peuvent entraîner le dysfonctionnement.
- On ne peut pas affecter une liste de plusieurs professeurs pour plusieurs matières sur plusieurs niveaux.
- On ne peut pas affecter plusieurs salles à une matière.
- On ne peut pas créer plusieurs sous-groupes d'une langue vivante ou d'une option à moins de créer une matière par sous-groupe.
- La contrainte sur le réfectoire ne fonctionne pas si un créneau horaire ne commence pas par "12h".
- Il faut impérativement écrire les matières ainsi que les professeurs avec la même orthographe sur toutes les parties.
- Il est possible que le volume horaire des professeurs soit dépassé, la contrainte n'est pas bloquante. Pour les salles, si une indisponibilité ponctuelle se présente pour placer un cours, le solveur utilisera "salle_inconnue", mais celle-ci ne figurera pas dans le fichier .JSON contenant les emplois du temps.

10. Glossaire

Voici les définition de l'ensemble des termes technique de ce présent document :

Terme	Définition
7. TER	Travail d'Étude et de Recherche, projet encadré réalisé en Master 1.
8. Dash	Framework Python permettant de créer des interfaces web interactives. Utilisé pour développer l'interface utilisateur du projet.

9. OR-Tools	Bibliothèque d'optimisation open-source développée par Google. Elle permet de modéliser et de résoudre des problèmes de contraintes.
10. Solveur	Composant logiciel qui modélise un problème à résoudre à partir de contraintes définies, et cherche des solutions optimales ou satisfaisantes.
11. JSON	(JavaScript Object Notation) Format léger et lisible pour structurer des données sous forme clé/valeur. Utilisé ici pour l'échange entre interface et solveur.
12. DataTable	Composant d'affichage de Dash permettant de visualiser ou de modifier des données en tableau.
13. Export PDF	Fonctionnalité permettant d'enregistrer les emplois du temps générés dans des formats lisibles et imprimables.
14. Initialisation (en programmation)	Action consistant à donner une valeur de départ à une variable ou un objet.
15. Boîte noire	Terme désignant une partie d'un système difficile à analyser ou à modifier sans en comprendre en détail le fonctionnement interne.
16. Interopérabilité	Capacité d'un système à échanger des données ou interagir avec un autre système de manière transparente.

11. Références

Cette section recense l'ensemble des documents produits ou utilisés dans le cadre du projet, ainsi que les ressources externes ayant contribué au développement, au suivi et à l'évaluation de l'application. Ces documents apportent un complément d'information utile pour comprendre les choix réalisés, le déroulement du projet et les modalités techniques.

- Documents internes au projet :
 - Cahier des charges.
 - Plan de développement : Document décrivant l'architecture logicielle envisagée, le découpage en modules, la répartition des tâches entre les membres du groupe, ainsi que les choix technologiques.

- Diagramme de Gantt : Planning prévisionnel du projet, indiquant les phases, les jalons, les tâches affectées à chaque membre et les dates de livraison attendues.
- Plan de test : Ensemble des cas de test définis pour valider le bon fonctionnement des modules (interface et solveur) et la cohérence des résultats.
- Compte-rendus des réunions.
- Guide d'installation : Manuel détaillant les étapes d'installation de l'environnement, du projet et des dépendances, pour un poste utilisateur.
- Guide d'utilisation : Manuel utilisateur décrivant le fonctionnement de l'interface, la gestion des données, l'utilisation des fonctionnalités, et l'export des résultats.
- Documentation interne : Description des modules, explication des fonctions principales.
- Code source commenté.
- [Dépôt Git](#)
- Ressources externes utilisées :
 - [Documentation officielle de Python.](#)
 - [Documentation Dash \(Plotly\).](#)
 - [Documentation OR-Tools \(Google\).](#)

12. Index

Voici la liste des mots-clés du document et où les trouver dans celui-ci :

Terme	Définition
1. TER	Page 4
2. Dash	Page 5/7/8/15
3. OR-Tools	Page 5/7
4. Solveur	Page 4/5/6/7/8/9/10
5. JSON	Page 7/8
6. DataTable	Page 8
7. Export PDF	Page 7

8. Initialisation (en programmation)	Page 8
9. Boîte noire	Page 9
10. Interopérabilité	Page 9

13. Annexes

Voici le diagramme de Gantt du second semestre :

