

Résumés en Français et en Anglais

*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

Résumés en Français et en Anglais

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 Résumé Français / Anglais	Validé par :	Aurélie Landry
Version document :	du 1.0	Validé le :	17/06/2025
Date du document :	12/06/2025	Soumis le :	12/06/2025
Auteur(s) :	Arbaut Jean-Baptiste Deschamps Kylian Duez-Faurie Valentine Eramil Kadir Pilloud Aubry Tropel Célia	Type de diffusion :	Document électronique (.pdf)
Les éléments d'authentification :		Confidentialité :	/
Maître d'ouvrage:	Aurélie Landry	Chef de projet :	/
Date / Signature :	17/06/2025 	Date / Signature :	/

1. Résumé en Français

Ce projet a été réalisé au cours de notre année de Master 1 MIASHS parcours Informatique et Cognition à l'Université Grenoble Alpes, dans le cadre de notre projet TER 2024-2025.

Initialement, l'outil devait être utilisé par un collègue avec lequel nous étions en lien, qui nous avait transmis des données d'exemple. Cependant, après analyse, il s'est avéré que le collègue utilisait déjà un logiciel très complet, Monoposte (Index Education), intégrant un générateur d'emploi du temps mais aussi un gestionnaire de planning ainsi que d'autres fonctionnalités comme la gestion des absences, des menus, de la messagerie, etc... C'est un outil d'établissement complet. Le collègue ne souhaitait donc pas se détacher d'un logiciel aussi complet.

Nous avons alors choisi, en collaboration avec notre maître d'ouvrage, de nous détacher de la demande initiale pour recentrer le projet sur : concevoir un générateur d'emploi du temps simple, avec à l'origine un objectif de permettre à l'utilisateur de créer de nouvelles contraintes. Nous avons ainsi conservé un périmètre clair, concentré sur la génération automatique d'emplois du temps classiques, tout en assurant une bonne ergonomie et une expérience utilisateur accessible.

Nous avons donc conçu et développé une application web dédiée à la génération automatique d'emploi du temps hebdomadaire pour les collèges. L'objectif étant de faciliter une tâche à l'origine longue, complexe et fastidieuse. Cette tâche est automatisée grâce à un système qui prend en compte l'ensemble des contraintes propres à chaque établissement et qui propose des emplois du temps dits optimisés, en respectant strictement les contraintes obligatoires (ex. : indisponibilités, capacités de salles) et en hiérarchisant les contraintes optionnelles (ex: préférences des professeurs pour une salle, poids des sacs,...) selon l'importance accordée par l'établissement.

L'application est composée de deux modules :

- Un solveur développé en Python, s'appuyant sur la bibliothèque OR-Tools de Google qui applique des techniques de recherche opérationnelle.
Il lit les données entrées dans l'interface graphique à partir d'un fichier de configuration JSON, modélise les contraintes (pédagogiques, humaines, matérielles, temporelles,...) grâce à des fonctions et les applique aux données. Les premiers cours sont définis de manière aléatoires puis sont définis en respectant les contraintes précédentes, il est donc possible de générer une infinité de solutions avec les mêmes données et les mêmes contraintes. Il compare un nombre défini d'emploi du temps et sélectionne la plus satisfaisante à l'aide de divers critères de satisfaction des contraintes comme les pourcentages de contraintes respectées ou lorsque l'ordre d'importance des contraintes est le plus respecté.
- Une interface web développée avec Dash (framework Python), qui permet de rendre

l'outil accessible à des utilisateurs non spécialistes en informatique. Elle permet de renseigner le programme, les ressources de l'établissement, les contraintes obligatoires des ressources (salles, professeurs, matières, temporelles...), les contraintes optionnelles (préférences des professeurs, indisponibilités partielles des ressources, heures de permanence,...) et une contrainte ayant pour objectif de préserver la santé des élèves en limitant le poids de leur sac par jour.

Il est ensuite possible de classer par ordre d'importance les contraintes non-obligatoires puis de lancer le générateur et visualiser les résultats obtenus ainsi que les statistiques concernant les emplois du temps obtenus et quelles contraintes n'ont pas pu être respectées.

Pour finir, il est possible d'ajuster manuellement certains cours pour répondre au mieux aux attentes.

Lorsqu'une version satisfaisante de l'emploi du temps est obtenue, il peut être exporté en pdf pour faciliter sa distribution.

L'objectif de l'interface est d'être la plus transparente et la plus intuitive possible. Pour cela les contraintes ont des formats prédéfinis et l'utilisateur doit les respecter, dans le cas contraire, il n'est pas possible d'entrer les données. L'utilisateur est guidé au fil des pages grâce à des explications concernant les données à entrer ainsi que leur utilité.

L'interface est simplifiée et la plus ergonomique possible. Ces deux choix permettent d'assurer une bonne expérience utilisateur grâce à l'ergonomie ainsi qu'au temps de prise en main de l'outil minime du fait des explications qui permettent de guider l'utilisateur au fil des pages.

Une série de tests unitaires a été mise en place pour garantir la conformité, la fiabilité et la régularité des solutions générées. Pour cela des exécutions aléatoires répétées ont été réalisées sur plusieurs itérations afin de détecter les incohérences. Pour chaque test, 50 itérations sont effectuées en appliquant une contrainte spécifique à tester suivie d'une vérification qu'aucun conflit n'est détectée et que la contrainte est bien respectée.

Ces tests ont permis de valider les contraintes, telles que l'unicité des affectations par créneau (un seul cours par salle, par enseignant et par classe), l'absence de cours pendant les horaires de cantine, ou encore le respect des indisponibilités et préférences définies dans les données de configuration.

L'ensemble du projet est accompagné d'une documentation complète : un cahier des charges, un cahier de recettes, un plan de développement, un manuel d'utilisation, un manuel d'installation, un plan de tests, une documentation interne, le rapport du projet, une vidéo de démonstration et l'intégralité des comptes rendus de réunion.

Ce générateur d'emplois du temps vise à offrir une solution efficace, modulable et évolutive répondant aux besoins spécifiques à chaque établissement scolaire.

2. Résumé en Anglais

This project has been realized during our first year of the Master's program in MIASHS – Computer Science and Cognition at Université Grenoble Alpes, as part of our 2024–2025 TER (Research and Study Project).

Initially, the tool was intended to be used by a partner middle school, which had provided us with sample data from the previous academic year. However, after analysis, we discovered that the school was already using a very comprehensive software solution, Monoposte (Index Éducation), integrating a schedule generator as well as a planning manager and other functions such as absence management, menus, messaging, and more. It is a full school management system. The school had no intention of switching from such a complete solution.

In agreement with our project sponsor, we therefore decided to move away from the original request and instead focus on developing a simple schedule generator, initially with the goal of allowing users to define their own constraints. We chose to keep a clear and realistic scope, focusing on the automatic generation of standard school schedules, while ensuring ease of use and a user-friendly experience.

We thus designed and developed a web application dedicated to the automatic generation of weekly school schedules for middle schools. The goal is to simplify a task that is typically long, complex, and tedious. This task is automated through a system that takes into account all the constraints specific to each institution and produces optimized schedules by strictly enforcing mandatory constraints (e.g., teacher availability, room capacity) and prioritizing optional constraints (e.g., teacher preferences for certain rooms, reducing backpack weight) based on the importance defined by the school.

The application is composed of two main modules:

- A solver developed in Python, using Google's OR-Tools library, which applies techniques from operations research.
It reads data entered via the graphical interface from a JSON configuration file, models the constraints (pedagogical, human, material, temporal, etc.) using dedicated functions, and applies them to the dataset. The first course assignments are generated randomly, then adjusted to respect all the defined constraints. As a result, an infinite number of valid solutions can be generated from the same data and constraints. The solver compares a set number of possible schedules and selects the most satisfactory one using various evaluation metrics such as the percentage of respected constraints or the best match with the defined priority order of the constraints.
- A web interface developed with Dash (a Python framework), designed to make the tool accessible to non-technical users. It allows users to input the academic program, the institution's resources (rooms, teachers, subjects), mandatory constraints (e.g.,

availability, time slots), optional constraints (e.g., partial unavailability, free periods, teacher preferences), and even a constraint aimed at preserving students' health by limiting daily backpack weight.

Users can rank the optional constraints by importance, run the generator, and then view the generated schedules and related statistics, including which constraints could not be satisfied.

Users can also manually adjust individual lessons to better fit specific needs. Once a satisfactory schedule is obtained, it can be exported as a PDF for easy distribution.

The interface is designed to be as transparent and intuitive as possible. All constraints must follow predefined formats; otherwise, the system will reject the data. The user is guided step by step through each screen, with clear explanations of the required inputs and their purposes.

The interface is streamlined and built with a focus on ergonomics and accessibility. These choices ensure a good user experience and a minimal learning curve, thanks to helpful guidance provided throughout the interface.

A series of unit tests was implemented to ensure the consistency, reliability, and accuracy of the generated solutions. To achieve this, we performed repeated random executions across multiple iterations to detect any inconsistencies. Each test consisted of 50 iterations where a specific constraint was applied and then verified to ensure that no conflicts were present and the constraint was respected.

These tests helped validate key constraints, such as unique assignments per time slot (only one class per room, teacher, and class at any given time), no classes during lunch hours, and the proper enforcement of defined availabilities and preferences.

The entire project is accompanied by comprehensive documentation, including a requirements specification, a test plan, a development plan, a user manual, an installation guide, an internal technical document, the final project report, a demonstration video, and all project meeting minutes.

This schedule generator aims to provide an efficient, modular, and scalable solution that can meet the specific planning needs of any educational institution.