Plan de développement

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

Plan de développement

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

Les informations d'identification du document :

Référence du document :	
Version du document :	1.01
Date du document :	12/01/2025
Auteur(s) :	Arbaut Jean-Baptiste Deschamps Kylian Duez-Faurie Valentine Eramil Kadir Pilloud Aubry Tropel Célia

Les éléments de vérification du document :

Validé par :	
Validé le :	
Soumis le :	12/01/2025
Type de diffusion :	Document électronique (.pdf)
Confidentialité :	

Les éléments d'authentification :

Maître d'ouvrage:	**Chef de projet :**
Date / Signature :	**Date / Signature :**

Sommaire

1. Introduction

L'objectif principal de ce projet est de concevoir et de développer un outil permettant la génération automatisée d'emplois du temps pour les établissements scolaires, tout en répondant à des contraintes multiples (fortes, moyennes ou faibles) et en offrant une flexibilité pour tester différents scénarios. Ce projet est réalisé dans le cadre d'un travail de recherche encadré (TER) par des étudiants de Master MIASHS à l'Université Grenoble Alpes.

L'outil devra répondre aux besoins exprimés par le commanditaire, Mme Landry, et par un utilisateur clé, M. Laffond, directeur de collège, tout en garantissant une conformité aux normes éducatives et au RGPD.

1.1 Objectifs et méthodes

Le développement du logiciel vise à concevoir une solution performante et adaptée pour la génération automatisée d'emplois du temps dans un collège, tout en répondant aux contraintes complexes exprimées par les parties prenantes. Ce projet repose sur une méthodologie Agile, permettant des itérations rapides et des ajustements constants en fonction des retours des utilisateurs et des encadrants.

Les principales étapes de réalisation comprennent :

- Analyse des besoins : Identification des contraintes et attentes des utilisateurs finaux.
- Conception : Développement de l'architecture logicielle et des interfaces utilisateur.
- Développement : Intégration des fonctionnalités principales, notamment l'implémentation du solveur OR-Tools.
- Tests et validation : Validation technique et fonctionnelle à travers des scénarios de tests rigoureux.
- Livraison : Fourniture d'un logiciel complet, accompagné de sa documentation utilisateur et technique.

Cette organisation méthodique assure que chaque phase du projet contribue à la réalisation d'un produit final fiable, performant et conforme aux attentes.

1.2 Documents de référence

Le présent document s'appuie sur un ensemble de ressources clés, essentielles à la conception et au développement du projet. Ces documents ont servi de base pour identifier les besoins, définir les contraintes, et structurer le déroulement du projet. Les documents de référence sont les suivants :

- Cahier des charges (Version 2.02 12.01.2025) : Document de référence décrivant les besoins fonctionnels, les contraintes techniques, et les objectifs du projet.
- Cahier de recette (Version 1.02 12.01.2025) : Document listant les scénarios de tests, les critères de validation, et les modalités de vérification de la qualité du logiciel.
- Documentation technique OR-Tools : Fournit les spécifications et les détails d'implémentation de l'algorithme utilisé pour gérer les contraintes complexes et optimiser les solutions.
- Exemples de données fournies par M. Laffond : Ensemble de données anonymisées permettant de tester et valider le logiciel dans des conditions réalistes.
- Documentation Monoposte : Référentiel pour assurer la compatibilité des fichiers exportés avec le logiciel de gestion d'emplois du temps Monoposte.

Ces documents garantissent une traçabilité et une cohérence dans toutes les étapes du projet, de sa conception à sa livraison.

2. Guide de lecture

Cette partie à pour objectif de guider les différentes parties prenantes dans l'utilisation optimale des informations fournies dans ce document. Elle détaille les objectifs et les méthodes d'utilisation pour les deux principaux groupes concernés : la maîtrise d'œuvre et la maîtrise d'ouvrage.

2.1 Maîtrise d'œuvre

Les équipes techniques responsables de la conception et du développement de l'outil peuvent utiliser ce document pour :

- Suivre les étapes du projet et garantir la conformité technique des livrables aux exigences définies.
- Identifier et résoudre les anomalies ou problèmes rencontrés lors des tests.
- S'appuyer sur les scénarios de tests et les critères de validation pour vérifier la qualité du produit à chaque phase.

2.2 Maîtrise d'ouvrage

Les décideurs et parties prenantes, en charge de valider les aspects fonctionnels et stratégiques du projet, utiliseront ce document pour :

- Vérifier que les livrables répondent aux besoins exprimés dans le cahier des charges.
- Participer aux phases de validation, notamment via les tests utilisateurs.
- Suivre les résultats des tests et des indicateurs de qualité pour évaluer la pertinence des solutions proposées.

3. Concepts de base

Pour comprendre ce document et les choix techniques réalisés dans le cadre de ce projet, il est essentiel de maîtriser certains concepts clés relatifs à la génération d'emplois du temps et à la gestion des contraintes. Cette section présente les notions fondamentales.

3.1 Contraintes fortes et faibles

Les *contraintes fortes* sont les conditions incontournables qui *doivent impérativement être respectées* pour qu'un emploi du temps soit valide. Elles incluent, par exemple, les indisponibilités des enseignants, les capacités des salles, et le respect du nombre d'heures par matière conformément aux programmes scolaires.

Les *contraintes faibles* sont des préférences ou des optimisations souhaitées qui peuvent être partiellement satisfaites en fonction des priorités définies. Par exemple, réduire les heures de permanence des élèves ou privilégier des cours le matin.

3.2 Solveur

Le *solveur* est un *algorithme* spécialisé dans la résolution de problèmes complexes sous contraintes. Il est capable d'optimiser les solutions en fonction des priorités définies. Le projet utilise *OR-Tools*, un solveur *open source* développé par *Google*, réputé pour sa robustesse et son efficacité dans la gestion de contraintes complexes.

3.3 Base de données

L'outil s'appuie sur une b*ase de données* conçue pour stocker toutes les informations nécessaires à la génération des emplois du temps, notamment :

- Les listes de professeurs, de classes, et de salles.
- Les contraintes spécifiques associées à chaque ressource.
- Les configurations des scénarios d'emplois du temps.

Cette architecture permet une gestion dynamique des données et offre la flexibilité requise pour tester différents scénarios selon les besoins des utilisateurs.

3.4 Compatibilité avec les outils tiers

Pour garantir une intégration fluide dans les systèmes existants, les emplois du temps générés peuvent être *exportés* au format *".edt"*, compatible avec des *logiciels comme Monoposte*. Cette compatibilité permet aux établissements scolaires de continuer à utiliser leurs outils habituels tout en bénéficiant de l'optimisation apportée par cet outil.

4. Organisation

L'organisation du projet repose sur une structuration claire des tâches et des équipes afin de garantir une réalisation efficace et coordonnée. Cette section détaille la décomposition en tâches ainsi que la structure des équipes impliquées dans le projet.

4.1 Décomposition en tâches

Le projet est découpé en plusieurs phases, chacune comprenant des tâches spécifiques. Voici la décomposition générale :

1. **Analyse des besoins :**
    - Identification des contraintes fortes et faibles.
    - Collecter les attentes des utilisateurs finaux et des parties prenantes.
2. **Développement :**
    - Développement de l'architecture de la base de données.
    - Création des interfaces utilisateurs (gestion des contraintes, visualisation).
    - Implémentation des fonctionnalités principales (solveur, gestion des données).
    - Intégration du solveur OR-Tools pour la génération et l'optimisation des emplois du temps.
3. **Tests et validation :**
    - Tests fonctionnels, d'intégration et de performance.
    - Tests utilisateurs pour évaluer l'ergonomie et la pertinence des solutions générées.
4. **Livraison :**
    - Préparation de l'exécutable final et des fichiers de configuration.
    - Rédaction de la documentation utilisateur et technique.

4.2 Structure des équipes

Pour mener à bien ce projet, voici les rôles et responsabilités que nous avons définis :

- Équipe de développement : Responsables de la conception, du codage, et de l'intégration des différentes fonctionnalités, représentée par les étudiants.

- Encadrants académiques : Supervisent le projet, valident les étapes clés et orientent les choix techniques.
- Utilisateur clé : Fournit les données nécessaires aux tests, valide les solutions proposées et participe aux sessions de tests utilisateur, représenté par M. Laffond.

Cette organisation hiérarchique et fonctionnelle permet une répartition claire des responsabilités et facilite la collaboration entre les différentes parties prenantes.

5. Planification

Le projet est organisé en quatres parties, certaines sont dépendantes les unes des autres et d'autres peuvent être réalisées en parallèle. dans cette partie nous allons détailler les différentes phrases, puis nous verrons le Gantt relatif au second semestre de notre année scolaire.

5.1 Organisation et préparation des outils collaboratifs

Cette phrase est primordiale, sa durée est fixée à tout le long du projet car en cas de retard, de problèmes, de difficultés, etc. nous devons être en capacité de répondre vite à notre problème avec une nouvelle organisation de travail.

- **Mise à jour de la définition de l'organisation du projet.**
- **Mise à jour des outils existants** :
    - Mise à jour du diagramme de Gantt pour intégrer les nouvelles échéances et assignations.
    - Mise à jour du tableau Trello pour suivre l'avancement des tâches.
    - Mise à jour du dépôt GitHub pour assurer une centralisation des documents (spécifications, diagrammes).

5.2 Développement de la base de données

- **Poursuite de la montée en compétence sur PostgreSQL.**
- **Conception de la base de données** :
    - Définition de la structure de la base (base de données relationnelle : entités, relations, contraintes).
    - Création et mise en œuvre de la base initiale.
    - Ajustements et évolutions en fonction des tests de performance ou de structure.
- **Tests de la base de données** :
    - Rédaction de scénarios de test (par exemple, génération d'emplois du temps, insertion de données fictives).
    - Import de jeux de données pour valider la robustesse et la fiabilité de la base.
    - Documentation des résultats pour prévoir les optimisations nécessaires.

5.3 Intégration du solver

- **Poursuite de la montée en compétence sur le solver** OR-Tools (Google).
- **Adaptation du solver aux contraintes spécifiques** :
    - Traduction des contraintes d'emploi du temps en un format exploitable par le solver (contraintes fortes, puis faibles).
    - Ajustements et évolutions en fonction des tests de performance ou de structure.
- **Tests du solver :**
    - Définition des scénarios de tests, réalisation des tests et documentation de ses derniers.

5.4 Développement des interfaces utilisateur

L'interface utilisateur constitue un élément central du projet, permettant aux utilisateurs finaux (enseignants, administrateurs) d'interagir avec l'outil de manière intuitive et efficace. Cette partie est divisée en plusieurs sous-étapes.

5.4.1 Conception de l'interface utilisateur

- **Étude des besoins utilisateurs :**
    - Analyse des besoins fonctionnels et ergonomiques : la collecte des retours utilisateurs que nous avons réalisée grâce à M. Laffond et notre analyse de l'outil.
    - Conception de maquettes pour visualiser les écrans principaux (création de l'emploi du temps, réglage des contraintes).

5.4.2 Développement de l'interface avec Dash

- **Création des composants principaux :**
    - Mise en place des écrans principaux en utilisant Dash :
        - Page de connexion.
        - Page d'accueil.
        - Page de saisie des données.
        - Page de création d'emplois du temps (interface d'ajout des enseignants, classes, matières, salles, etc.).
        - Page de visualisation des emplois du temps générés.

- **Connexion avec la base de données :**
    - Intégration des données dynamiques issues de la base (affichage et modification en temps réel).

5.4.3 Tests et amélioration de l'interface

- **Tests utilisateurs :**
    - Organisation de sessions de tests pour vérifier la fluidité, la clarté et l'ergonomie.
- **Corrections et ajustements :**
    - Amélioration des éléments identifiés comme problématiques (par exemple, la navigation ou la vitesse de chargement).

5.5 Tests globaux et finalisation

- **Intégration finale** :
    - Validation de l'interaction entre la base de données et le solver.
    - Test des fonctionnalités principales (génération automatique d'emplois du temps, gestion des conflits).
- **Documentation finale** :
    - Création de manuels utilisateurs et documents techniques pour faciliter la maintenance et la prise en main.
    - Réalisation du rapport du projet et de la présentation pour la soutenance.

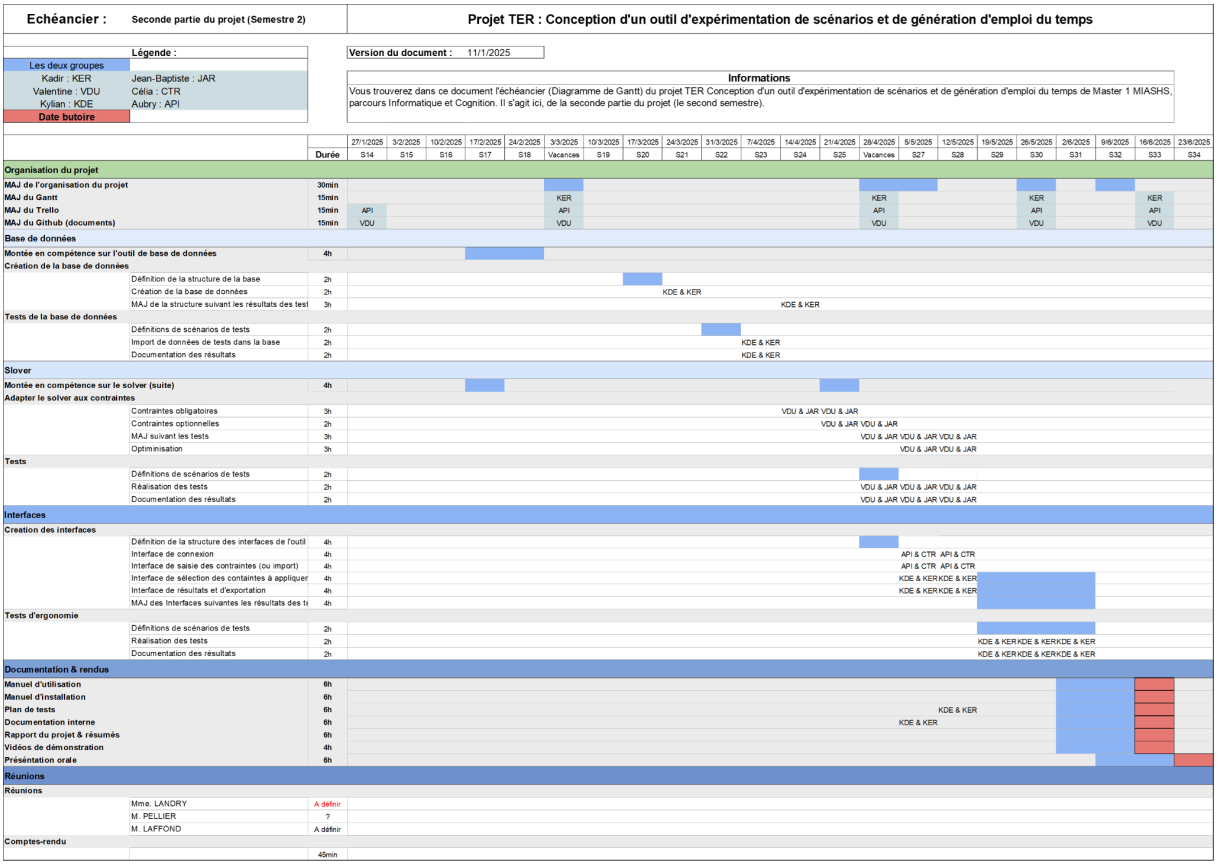
5.6 Calendrier prévisionnel des parties

1. **Janvier 2025 :**
    - Organisation du projet : mise à jour des outils collaboratifs.
    - Poursuite de la montée en compétence sur la base de données (et le solver).
2. **Février 2025 :**
    - Définition et création de la base de données.
    - Premiers tests de la base.
3. **Mars 2025 :**
    - Finalisation des tests de la base de données.
    - Documentation des résultats et ajustements nécessaires.
4. **Avril 2025 :**
    - Première phase de développement du solver.

- Première phase de création d’interfaces.

5. **Mai 2025 :**
   - Intégration complète du solver aux contraintes.
   - Tests globaux et corrections.
   - Poursuite de la création des interfaces et tests d’ergonomie.
6. **Juin 2025 :**
   - Finalisation du projet, tests d’ensemble, et livraison.
   - Documentation utilisateur et technique.

5.7 Représentation sous forme de diagramme de Gantt



6. Cycle de vie

Le cycle de vie du projet suit un modèle structuré en plusieurs phases afin de garantir une progression méthodique et maîtrisée. Chaque phase est conçue pour répondre à des objectifs spécifiques tout en assurant une validation continue des livrables. Ces phases s'appuient sur des méthodes de gestion de projets que nous avons adapté à notre projet, son organisation et disponibilités vis-à-vis de nos cours.

6.1 Phases du Cycle de Vie

1. **Phase d'initiation :**
    - Identification des parties prenantes et définition des objectifs globaux du projet.
    - Élaboration des documents de base, tels que le cahier des charges et le cahier de recette.
2. **Phase de conception :**
    - Création de l'architecture logicielle, y compris la base de données et les interfaces utilisateur.
    - Sélection des outils et technologies (OR-Tools, format ".edt", etc.).
3. **Phase de développement :**
    - Implémentation des fonctionnalités principales, comme la gestion des contraintes et la génération des emplois du temps.
    - Intégration des modules (solveur, base de données, interfaces).
4. **Phase de tests et validation :**
    - Réalisation de tests unitaires, d'intégration, et fonctionnels.
    - Validation des résultats avec les utilisateurs tests.
5. **Phase de livraison :**
    - Préparation et fourniture de l'exécutable final, accompagné de la documentation utilisateur et technique.
    - Formation et accompagnement des utilisateurs finaux pour la prise en main de l'outil.
6. **Phase de maintenance et évolutions :**
    - Suivi des retours utilisateurs pour corriger les éventuels bugs ou améliorer les fonctionnalités.
    - Intégration des nouvelles contraintes ou besoins identifiés après la mise en production.

6.2 Itérations et Validation

Le cycle de vie intègre une approche itérative, permettant des livraisons progressives et des ajustements constants en fonction des retours des parties prenantes. Chaque phase se termine par une validation formelle des livrables, garantissant la continuité et la qualité du projet.

7. Méthodes et outils

Le projet s'appuie sur une méthodologie structurée et des outils performants pour garantir une exécution alignée sur les objectifs fixés. Cette section décrit les méthodes adoptées ainsi que les outils utilisés tout au long du cycle de vie du projet.

7.1 Méthodes

1. **Méthodologie Agile :**
    - Utilisation d'une approche itérative pour permettre des livraisons progressives et des ajustements constants en fonction des retours des utilisateurs.
    - Organisation en sprints définis par des étapes clés comme l'analyse, la conception, le développement et les tests.
2. **Développement orienté sur les besoins :**
    - Mise en place de cycles d'analyse et de validation réguliers avec les parties prenantes pour s'assurer que les fonctionnalités développées répondent aux attentes exprimées.
    - Focus sur la gestion des contraintes (fortes et faibles) pour maximiser la pertinence des solutions générées.
3. **Gestion de projet :**
    - Utilisation d'un planning détaillé (incluant un diagramme de Gantt) pour suivre les délais et répartir les responsabilités.
    - Suivi des livrables à travers des jalons définis pour chaque phase.

7.2 Outils

- **Outils de développement :**
  - OR-Tools : Solveur utilisé pour gérer les contraintes complexes et optimiser les emplois du temps générés.
  - Environnement de développement intégré (IDE) : Visual Studio Code ou un équivalent pour le codage et la gestion du projet.
- **Outils de gestion de version :**
  - GitHub : Plateforme utilisée pour stocker le code source, collaborer entre membres de l'équipe, et gérer les versions du logiciel.
- **Base de données :**
  - Mise en place d'une base de données structurée pour stocker les informations nécessaires, comme les contraintes, les ressources (enseignants, salles, classes), et les scénarios.
- **Outils de documentation :**
  - **Microsoft Word / Google Doc** : Pour la rédaction de la documentation technique et utilisateur.
  - **Outils de gestion de planning** : Trello et diagramme de Gantt pour organiser et suivre les tâches du projet.

8. Documentation

La documentation joue un rôle essentiel dans notre projet, en assurant une compréhension claire et une prise en main facile de l'outil par les utilisateurs finaux, ainsi qu'une maintenance efficace par des développeurs. Cette section présente la structure de la documentation ainsi que les standards et outils utilisés pour sa rédaction.

8.1 Présentation

La documentation est divisée en deux principales catégories :

- **La documentation utilisateur :**
  - Qui devra fournir des instructions détaillées pour installer, configurer et utiliser l'application.
  - Qui devra contenir des exemples illustrés pour guider les utilisateurs à travers les différentes fonctionnalités, comme la saisie des contraintes ou l'exportation des emplois du temps.
  - Qui devra être livrée sous la forme d'un manuel au format PDF (et intégrée, si possible, dans l'interface de l'application).
- **La documentation technique :**
  - Qui devra décrire l'architecture logicielle, y compris la structure de la base de données, le fonctionnement du solveur OR-Tools, et les flux de données entre les différents modules.
  - Qui devra inclure des instructions pour les développeurs souhaitant maintenir ou étendre l'application.
  - Qui devra fournir des détails sur les interfaces d'intégration et les formats d'échange de données (notamment le format ".edt").

8.2 Standards et outils

Voici les standards et outils pour la documentation du projet :

- **Standards :**
  - Suivi des conventions de documentation technique pour garantir une lisibilité et une uniformité.
  - Utilisation d'une nomenclature claire pour nommer les variables, modules, et bases de données dans la documentation.
  - Respect des directives RGPD dans les exemples pour anonymiser les données sensibles.
- **Outils utilisés :**
  - Microsoft Word / Google Docs : Rédaction et mise en page des documents.
  - GitHub Wiki : Documentation technique pour les développeurs, intégrée au dépôt GitHub.
  - Outils de diagrammes / schémas : Création de schémas pour illustrer les architectures, les flux de données, et les processus.

9. Qualité

La gestion de la qualité est un aspect fondamental de ce projet. Elle garantit que le produit final répond aux exigences fonctionnelles et techniques, tout en assurant une expérience utilisateur optimale. Cette section décrit les mesures mises en œuvre pour garantir un haut niveau de qualité tout au long du cycle de vie du projet.

9.1 Objectifs de qualité

L'objectif principal en matière de qualité est de fournir un outil performant et fiable, capable de générer des emplois du temps respectant les contraintes complexes, tout en offrant une interface utilisateur intuitive et accessible, adaptée aux besoins des différents profils d'utilisateurs. Le logiciel doit également garantir la compatibilité des fichiers exportés avec les outils tiers comme Monoposte. Par ailleurs, le projet respecte les exigences légales, notamment en matière de protection des données (RGPD), afin d'assurer la conformité et la sécurité des informations manipulées.

9.2 Stratégie de qualité

1. **Les tests :**
    - Tests fonctionnels : Validation de chaque fonctionnalité, comme la saisie des contraintes, la génération des emplois du temps, et l'exportation des fichiers.
    - Tests d'intégration : Vérification de l'interopérabilité entre les différents modules (solveur, base de données, interfaces).
    - Tests de performance : Mesure des temps de calcul pour s'assurer de la réactivité du solveur, même avec des volumes importants de données.
    - Tests utilisateurs : Sessions dédiées pour évaluer l'ergonomie et la pertinence des résultats.
2. **Validation progressive :**
    - Validation des livrables à chaque phase du projet, notamment après l'analyse, la conception, et le développement.
    - Collaboration étroite avec les parties prenantes pour s'assurer que les besoins exprimés sont satisfaits.
3. **Indicateurs de qualité :**
    - Taux de satisfaction des contraintes fortes et faibles.

- Pourcentage de complétion des emplois du temps générés.
- Feedback des utilisateurs finaux sur l'ergonomie et la simplicité d'utilisation de l'outil.

9.3 Gestion des non-conformités

Pour la gestion des non-conformités ou d'anomalies identifiées lors des tests ou de l'utilisation de l'outil :

- Les anomalies seront documentées et analysées pour identifier leur origine.
- Des correctifs seront priorisés et intégrés dans des cycles itératifs de développement.
- Les solutions corrigées seront soumises à de nouveaux tests pour garantir leur fiabilité.

10. Glossaire

Agile : Méthodologie de gestion de projet favorisant des cycles itératifs et des ajustements continus en fonction des retours utilisateurs.

Base de données relationnelle : Système structuré pour stocker et gérer des données, organisé en tables avec relations et contraintes.

Cahier des charges : Document détaillant les exigences fonctionnelles et techniques du projet.

Cahier de recette : Document définissant les scénarios de tests, les critères de validation et les modalités de vérification du projet.

Contraintes fortes : Conditions indispensables devant être respectées, telles que les indisponibilités des enseignants et la capacité des salles.

Contraintes faibles : Préférences ou optimisations souhaitées, comme la réduction des heures de permanence des élèves.

Dash : Framework Python utilisé pour créer des interfaces web interactives et des tableaux de bord.

Diagramme de Gantt : Représentation graphique d'un calendrier de projet, montrant les différentes tâches et leur chronologie.

Emploi du temps (EDT) : Organisation temporelle des activités scolaires (cours, pauses, etc.) pour un établissement.

Monoposte : Logiciel tiers pour la gestion des emplois du temps compatible avec les fichiers générés par le projet.

OR-Tools : Solveur open source développé par Google, utilisé pour résoudre des problèmes complexes sous contraintes.

RGPD : Règlement Général sur la Protection des Données, encadrant la collecte, le traitement et le stockage des données personnelles.

Solveur : Algorithme conçu pour résoudre des problèmes complexes, comme la planification sous contraintes.

Sprint : Cycle court dans une méthodologie Agile, avec des objectifs spécifiques pour chaque phase de développement.

TER (Travail Encadré de Recherche) : Projet académique réalisé dans un cadre universitaire, souvent à titre exploratoire ou pratique.

11. Références

Les références bibliographiques apportant des informations complémentaires :

- Cahier des charges :
  - Version 2.02, datée du 12 janvier 2025.
  - Document définissant les besoins fonctionnels, les contraintes techniques, et les objectifs du projet.
- Cahier de recette :
  - Version 1.02, datée du 12 janvier 2025.
  - Document contenant les scénarios de tests, les critères de validation, et les modalités de vérification de la qualité du logiciel.
- Documentation technique OR-Tools : Ressource officielle pour l'implémentation et l'utilisation du solveur OR-Tools.
- Exemples de données fournies par M. Laffond : Ensemble de données anonymisées pour tester et valider le logiciel dans des conditions réalistes (disponible en annexe du cahier des charges).
- Documentation Monoposte : Référentiel utilisé pour assurer la compatibilité des fichiers générés au format ".etd" avec le logiciel Monoposte.
- Outils collaboratifs utilisés :
  - GitHub : Dépôt pour la gestion de version et la collaboration sur le code source du projet.
  - Diagramme de Gantt : Suivi des tâches et organisation.
  - Trello : outil de gestion de projet.

12. Index

Références	5, 24	Sommaire, Références
Scénarios de tests	5, 12, 15, 20	Documents de référence, Développement des interfaces, Cycle de vie, Qualité
Solveur	7, 15, 16, 23	Concepts de base, Cycle de vie, Méthodes, Glossaire
Sprint	16, 23	Méthodes, Glossaire
Standard	18	Documentation
Structure d'équipe	9	Organisation
Travail Encadré de Recherche	4, 15, 23	Introduction, Cycle de vie, Glossaire
Université Grenoble Alpes	4	Introduction
Utilisateur clé	9	Organisation
Validation	4, 15, 20	Objectifs et méthodes, Cycle de vie, Qualité