Cahier de recette

Projet : Conception d'un outil pour améliorer la productivité en préservant la limite du travail humain

BEHAR Andrea
MARTINIANI Anesie
M1 MIASHS- IC

Les informations d'identification du document : ***Les éléments de vérification du document :***

Référence du document :	**Validé par :**
Version du document : 1.02	**Validé le :**
Date du document : 12/01/2026	**Soumis le :** 12/01/2026
Auteur(s) : Andrea Behar Anesie Martiniani	**Type de diffusion :** Document électronique (.pdf)
	Confidentialité :

Les éléments d'authentification :

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

Sommaire

1. Introduction

Ce cahier de recette a pour objectif de garantir la conformité des fonctionnalités de l'outil pour améliorer la productivité de DPC en aux exigences définies dans le cahier des charges. Il structure les étapes de validation nécessaires, de la gestion des contraintes fortes et faibles.

Le projet repose sur une méthodologie Agile, permettant des livraisons itératives et des ajustements en fonction des retours des utilisateurs et dans notre cas, de la cheffe de projet Mme. Landry. Parmi les étapes clés figurent l'analyse approfondie des besoins, la conception d'une base de données, l'adaptation du solveur LPG pour gérer des contraintes complexes.

Le périmètre de la recette inclut la vérification des fonctionnalités telles que la gestion des contraintes et les performances du solveur.

1.1. Objectifs et méthodes

Le développement du logiciel a été structuré en plusieurs étapes clés afin de garantir une réalisation efficace et conforme aux exigences du projet. Ces étapes s'appuient donc sur une méthodologie Agile, avec des livraisons progressives et des ajustements itératifs en fonction des retours des parties prenantes.

1. Analyse des besoins :

Cette partie à été réalisée durant la première phase du projet, en parallèle de la réalisation du cahier des charges. Nous avions :

- L'étude des besoins : Analyse approfondie des attentes exprimées par les parties prenantes.
- La définition des contraintes : Identification et formalisation des contraintes fortes, moyennes et faibles.
- La sélection de l'algorithme : Le solveur LPG, capable de gérer des contraintes complexes.

2. Conception de la base de données :

Conception de l'architecture de la base de données :

- Création d'une base de données pour gérer les informations relatives aux contraintes humaines et à l'amélioration de la productivité.
- Définition d'une architecture permettant d'évoluer par la suite en cas de changement.

3. Développement du solveur :

- Adaptation au projet : Adaptation de l'algorithme / solver pour notre projet et notre type de données.
- Optimisation des performances : Réduction des temps de calcul pour garantir une génération

rapide de l'organisation du processus de picking.

4. Documentation :

- Guide utilisateur : Création d'un manuel expliquant les étapes de prise en main, depuis l'installation jusqu'à l'exploitation des résultats.
- Documentation technique : Description détaillée de l'architecture du logiciel, des algorithmes utilisés, et des formats d'échange de données.

5. Validation et recette :

Cette partie sera détaillée par la suite.

- Tests unitaires : Vérification de chaque composant pour s'assurer de son bon fonctionnement.
- Tests d'intégration : Validation de l'interopérabilité entre les différentes parties du logiciel.
- Scénarios de tests : Utilisation de données réelles pour simuler des cas d'utilisation concrets et vérifier la conformité avec les exigences.

7. Livraison :

Cette partie sera détaillée par la suite.

1.2. Périmètre de la recette

Voici les fonctionnalités que nous devons tester :

1. Gestion des contraintes :

- Ajout, modification et suppression de contraintes dans la base de données.
- Gestion des contraintes fortes.
- Gestion des contraintes faibles.

2. Génération des plans logistiques :

- Prise en compte des contraintes saisies pour produire une solution.
- Affichage des différentes solutions proposées.

3. Performances et fiabilité :

- Capacité à traiter des volumes importants de données.
- Temps de calcul raisonnable pour la génération des solutions.

1.3. Documents de référence

Le cahier de recette s'appuie sur plusieurs ressources et documents essentiels pour son élaboration.

Voici la liste des références utilisées :

- Cahier des charges : Version 2.02, datée du 12/01/2026, décrivant les besoins fonctionnels et techniques ainsi que les contraintes liées au projet.

2.Guide de lecture

Le guide de lecture permet d'aider les lecteurs à utiliser efficacement ce document en fonction de leur rôle et de leurs responsabilités. Chaque sous-partie est adaptée aux besoins spécifiques des différents types de lecteurs, leur offrant des instructions claires et adaptées à leurs missions.

2.1.Maîtrise d'œuvre

La maîtrise d'œuvre intervient dans la gestion technique du projet. De ce fait, ce document leur permet de suivre scrupuleusement les différents tests prévus afin de garantir la conformité technique des livrables aux exigences définies. En cas d'anomalies ou de dysfonctionnements identifiés lors des tests, il est essentiel de les documenter avec précision dans le but d'assurer leur correction rapide et efficace. Ce processus rigoureux vise à maintenir la qualité et la fiabilité des solutions techniques déployées.

2.2.Maîtrise d'ouvrage

La maîtrise d'ouvrage intervient principalement pour s'assurer que les solutions proposées répondent parfaitement aux besoins exprimés. À travers ce document, les membres de l'équipe maîtrise d'ouvrage peuvent valider les scénarios de test pour vérifier leur alignement avec les attentes initiales. Ils participent également aux tests utilisateurs, un moment crucial pour évaluer l'ergonomie et la pertinence des fonctionnalités. Leur contribution garantit que les solutions livrées sont en adéquation avec les objectifs du projet.

3.Concepts de base

Dans cette partie nous allons préciser les concepts de base nécessaires à la compréhension du document. Notre projet a pour objectif la conception d'un outil pour optimiser l'organisation d'un processus de picking en logistique.

D'après nos échanges avec Mme. Laundry, cheffe de projet, chez DPC les employeurs doivent délivrer environ 600 commandes par jour, normalement réparties sur des agents traitant chacun environ 30 commandes par jour. Le processus logistique actuel est le suivant: réception des palettes hétérogènes (produits mélanges), stockage par tri et mise en stock en palettes homogènes dans l'entrepôt (système type IKEA), et finalement la préparation de commandes par picking d'articles variés pour créer des palettes hétérogènes selon les commandes clients. Selon DPC la productivité actuelle est jugée insuffisante et le côté ergonomie peut être amélioré pour éviter la quantité actuelle d'accidents. Notre projet va servir pour analyser la situation actuelle et va essayer de donner des suggestions pour optimiser l'ergonomie des employés en ligne avec les commandes qui sont demandées.

Pour répondre à notre demande, nous utiliserons un *algorithme*, appelé "*Solveur*", capable de trouver *des solutions* en fonction de *contraintes fortes et/ou faible*s. Les objectifs de l'outil seront de répondre à un maximum de contraintes pour chaque solution et de permettre de tester des *scénarios* de contraintes.

Notre outil utilisera des données provenant d'une *base de données*, que nous aurons conçue. Il est donc important d'identifier l'architecture de la base et les données nécessaires en amont.

4.Description de la fourniture

L'outil sera livré sous forme d'un programme exécutable compatible avec les systèmes d'exploitation les plus courants (Windows, Mac et Linux). Elle sera accompagnée des éléments suivants :

- La documentation utilisateur : Un guide utilisateur détaillé au format PDF, contenant des instructions sur l'installation, la configuration et l'utilisation de l'outil. Les étapes seront aussi détaillées.

- La documentation technique : Un document technique décrivant l'architecture du système, les principales fonctionnalités, les dépendances et les instructions pour les développeurs souhaitant maintenir ou faire évoluer l'outil.

5.Moyen d’essai et outils

Cette section décrit les moyens et outils mobilisés pour vérifier la conformité de l’outill avec les exigences spécifiées dans le cahier des charges.

Dans un premier temps, voici les outils utilisés :

- Un IDE (Environnement de Développement Intégré) pour la création et la maintenance du code source (exemple : Visual Studio Code).
- Une Plateforme de gestion de version (GitHub) : Utilisée pour centraliser le code, gérer les versions, et collaborer efficacement entre les membres de l'équipe.

Cette liste d'outils est susceptible d’évoluer suivant les retours des premières semaines de développement.

Dans un second temps, la validation de l’application repose sur une méthodologie structurée inspirée des pratiques Agile. Cette approche combine des tests itératifs à chaque étape de développement et une validation finale basée sur des scénarios réels.

Voici les tests permettant de vérifier la conformité :

- Tests fonctionnels :
  - Objectif : Valider chaque fonctionnalité indépendamment pour garantir qu’elle répond aux spécifications.
  - Exemples : Vérification de la saisie des contraintes, de l’exportation des fichiers “.edt”, de l’ajout de nouvelles données dans la base.
- Tests d’intégration :
  - Objectif : Vérifier l’interopérabilité entre les différents modules du logiciel (solveur, base de données).
  - Méthodologie : Test l’entièreté du processus pour s'assurer que les contraintes saisies sont bien intégrées au solveur et reflétées
- Tests utilisateurs :
  - Objectif : Recueillir les retours des utilisateurs testeurs sur l’ergonomie, la facilité d’utilisation, et la pertinence des résultats.
  - Méthodologie : Créer des sessions de tests, observer les utilisateurs testeurs et leur faire compléter un questionnaire post-test sur l’ergonomie de l’outil (navigation, difficultés rencontrées, etc.).
- Tests de performance :
  - Objectif : Vérifier que l’application est rapide et fluide, même avec des volumes

de données importants.

- Mesures à tester : Temps de calcul du solveur, temps de chargement des interfaces et capacité à gérer simultanément plusieurs scénarios de contraintes.

6.Conformité aux spécifications générales

Dans cette section, nous allons décrire les vérifications nécessaires pour s'assurer que l'application respecte les spécifications générales définies dans le cahier des charges. Nous préciserons les critères de validation et les méthodes employées pour évaluer la conformité du produit final.

1. Respect des contraintes fortes :

L'outil doit garantir que toutes les contraintes fortes soient respectées. Cette conformité sera vérifiée par une analyse systématique des solutions proposées pour s'assurer que ces contraintes sont appliquées sans exception.

2. Satisfaction des contraintes faibles :

Les contraintes faibles est un objectif requis. Cette exigence sera vérifiée par une évaluation des solutions générées, en mesurant le taux de satisfaction des préférences définies par les utilisateurs.

3. Performance de génération :

Les temps de calcul seront mesurés lors des tests de performance réalisés avec différents volumes de données pour valider cet objectif.

4. Indicateurs de qualité :

L'outil doit fournir des indicateurs clairs sur la qualité des solutions générées, incluant le pourcentage de complétion, le nombre de contraintes non respectées, et l'identification des contraintes restrictives.

5. Conformité légale et réglementaire :

L'outil doit respecter le Règlement Général sur la Protection des Données (RGPD) en garantissant l'anonymisation des données sensibles lors des tests et des démonstrations. Cette conformité sera vérifiée par un audit des données manipulées et une documentation claire des processus de protection des données.

8. Méthodes de vérification :

Pour valider les critères ci-dessus, les tests sont organisés en plusieurs phases :

- Les tests fonctionnels permettent de valider chaque fonctionnalité de manière isolée pour garantir qu'elle répond aux spécifications.
- Les tests d'intégration vérifient que les modules (solveur, base de données) interagissent correctement entre eux.
- Les tests de performance mesurent les temps de calcul et évaluent la robustesse du logiciel face à des volumes importants de données.
- Les tests utilisateurs recueillent les retours des parties prenantes sur l'expérience utilisateur et la pertinence des résultats.

7.Conformité aux spécifications fonctionnelles

Cette section décrit les scénarios de tests nécessaires pour s'assurer que l'application respecte le périmètre fonctionnel défini lors de la phase de spécification. Chaque scénario est présenté de manière détaillée pour valider les fonctionnalités clés de l'outil.

Les tests scénarios seront complétés lorsque nous aurons les données nécessaires à leur construction.

7.1. Scénario 01 : Gestion des contraintes fortes

Description : Ce test vise à garantir que les contraintes fortes sont correctement appliquées et respectées. Le test sera effectué dans un environnement simulé avec des données de test réalistes.

Contraintes : Nécessite des données définissant les contraintes fortes.

Dépendances : La base de données doit être correctement initialisée avec les informations.

Procédure de test :

Données en entrée : Interdit de faire X position ergonomique.

Résultat attendu : Toutes les solutions générées respectent intégralement la contrainte forte (obligatoire).

7.2. Scénario 02 : Gestion des contraintes faibles

Description : Ce test vérifie que les contraintes faibles sont optimisées dans les solutions proposées par le solveur, tout en respectant les contraintes fortes.

Contraintes : Nécessite des données définissant les contraintes faibles.

Dépendances : Les contraintes fortes doivent être validées avant ce test.

Procédure de test :

Données en entrée : Contrainte faible.

Résultat attendu :

7.3. Scénario 03 : Multi-solution pour un plan

Description :

Contraintes : Nécessite la saisie de contraintes avec des poids variables.

Dépendances :

Procédure de test :

Données en entrée : Ensemble de contraintes.

Résultat attendu : L'outil propose au moins deux solutions distinctes.

7.4. Scénario 04 : Modification post-génération

Description : Ce test vérifie que les contraintes peuvent être modifiées après une génération initiale et que le solveur peut produire une nouvelle solution adaptée.

Contraintes : Les modifications doivent être appliquées sur un plan existant.

Dépendances : Nécessite la fonctionnalité de gestion des contraintes et le solveur.

Procédure de test :

Données en entrée : Plan généré initialement et contrainte modifiée.

Résultat attendu : La nouvelle solution respecte la contrainte modifiée.

.

8. Glossaire

Algorithme : Suite d'instructions ou de règles logiques permettant de résoudre un problème donné, comme la génération d'emplois du temps dans ce projet.

Base de données : Structure organisée pour stocker et gérer les informations nécessaires à l'application, notamment les données sur les professeurs, les salles, les classes, et les contraintes.

Contraintes fortes : Conditions incontournables pour la génération d'un emploi du temps valide, telles que les indisponibilités des professeurs ou les capacités des salles. Leur respect est obligatoire pour chaque solution générée.

Contraintes faibles : Préférences ou optimisations souhaitées dans un emploi du temps, comme la réduction des heures de permanence. Elles ne sont pas obligatoires mais sont prises en compte en fonction de leur priorité.

Critères de Bastien & Scapin : Les critères ergonomiques de Bastien et Scapin sont des principes définis pour évaluer et concevoir des interfaces utilisateur de manière ergonomique dans le but d'améliorer l'expérience utilisateur. Ils sont regroupés en huit grandes catégories : Guidage, Charge de travail, Contrôle explicite, Adaptabilité, Gestion des erreurs, Homogénéité / Cohérence, Signifiance des codes / dénominations et Compatibilité. Voici un lien expliquant chaque catégorie.

Indicateurs de qualité : Mesures fournies par l'application pour évaluer la qualité des emplois du temps générés, incluant le pourcentage de complétion et le nombre de contraintes respectées.

LPG: Solveur open source utilisé pour gérer les contraintes complexes et proposer des solutions optimisées dans le cadre de ce projet.

RGPD (Règlement Général sur la Protection des Données) : Règlement européen visant à protéger les données personnelles. L'application doit garantir la conformité à ces règles en anonymisant les données sensibles.

Scénario de test : Suite d'étapes prédéfinies permettant de vérifier que l'application répond aux spécifications et satisfait les critères de validation.

Solveur : Algorithme spécialisé utilisé pour résoudre des problèmes sous contraintes. Il génère des solutions optimales en fonction des priorités définies.

Temps de calcul : Durée nécessaire au solveur pour générer un emploi du temps à partir des contraintes saisies.

Test fonctionnel : Test permettant de valider qu'une fonctionnalité de l'application fonctionne indépendamment des autres.

Test d'intégration : Test visant à vérifier le bon fonctionnement et l'interaction entre les différents modules de l'application.

Test de performance : Évaluation de la rapidité et de la fluidité de l'application, notamment la durée de calcul du solveur pour des volumes importants de données.

9. Références

Les références bibliographiques apportant des informations complémentaires :

- Cahier des charges :
    - Version 2.02, datée du 12 janvier 2026.
    - Document définissant les besoins fonctionnels, les contraintes techniques, et les objectifs du projet.

- Cahier de recette :

○ Version 1.02, datée du 12 janvier 2025.
○ Documents contenant les scénarios de tests, les critères de validation, et les modalités de vérification de la qualité du logiciel.

- Plan de Développement :

Version 1.01, datée du 12 janvier 2025.
○ Document décrivant l'organisation, les méthodes, et les outils utilisés pour la réalisation du projet.

- LPG : Spécifications et détails d'implémentation du solveur utilisé.
- PDDL4J's documentation! : Référentiel technique pour le solveur ILog CPLEX.

- Liens vers les outils collaboratifs du projet :

○ TER Andrea BEHAR & Anesie MARTINIANI : Documents centralisés du projet (Google Drive).
○ GitHub: Dépôt pour le code source et les contributions.