**KitchenCompanion**

*Application de gestion et partage de recettes*

**Document de conception**

Par :

Rémi Chuet - *2059171*

Julien Coulombe-Morency - *6103438*

Travail présenté à :

Jean-Christophe Demers

420-C61-IN : Projet synthèse

Cégep du Vieux Montréal

7 mars 2024

**Rappel projet**

Notre projet est de concevoir un système flexible et intuitif qui simplifie la formulation des recettes, uniformise les méthodes de travail et garantit un accès aisé à ces informations pour tous les membres d'une brigade.

Gestion des couts

Gestion des ressources humaines (brigades cuisines)

Gestion des connaissances métier (recettes, plats)

**Présentation**

Le document de conception joue un rôle crucial dans notre projet, visant à assurer la robustesse et la qualité du logiciel en développement. Son objectif principal est de faciliter la maintenance en fournissant un point de référence clair pour la compréhension du système tout au long du projet. En guidant les programmeurs, il évite les casse-têtes en découpant les tâches de manière logique et compréhensible. De plus, il clarifie la définition des fonctionnalités en expliquant comment certaines problématiques ou caractéristiques seront abordées. La communication efficace est également un objectif clé, garantissant que toutes les parties prenantes comprennent les spécifications et les objectifs des différentes parties du projet. Enfin, le document de conception vise à réduire les risques en identifiant et en définissant les défis potentiels dès le départ.

L’objectif de ce document est de :

Faciliter la maintenance (aider pour la compréhension et avoir un point de repère au courant du projet)

Guider les programmeurs (pas de casse-tête, une grande partie devrait être découpé)

Clarification de la définition (clarifier comment certaines problématique ou feature seront faite)

Communication efficace (que tous comprennent les spécifications et les objectifs des portions)

Réduire les risque (définir les défis potentiel)

**Infrastructure de développement**

**Plateforme cible** : Notre projet vise à être accessible sur toutes les plateformes, ce qui en fait une application web. Cette approche offre une portabilité maximale et permet à nos utilisateurs d'accéder à notre application depuis n'importe quel appareil disposant d'un navigateur web, qu'il s'agisse de PC, de tablettes ou de smartphones.

**Langage de programmation** : Nous avons opté pour TypeScript (TS) pour le développement de notre application. TypeScript offre les avantages d'un typage statique, ce qui facilite la détection des erreurs lors de la phase de développement et améliore la maintenabilité du code. De plus, en utilisant HTML et CSS, nous assurons une compatibilité maximale avec les navigateurs et une facilité de mise en page.

**Bibliothèques principales** : Pour le développement de l'interface utilisateur, nous avons choisi d'utiliser React pour sa popularité, sa flexibilité et sa richesse en fonctionnalités. Nous avons également opté pour Remix pour ses performances et sa facilité d'intégration avec React. En ce qui concerne la conception et le style, nous avons choisi Tailwind CSS pour sa facilité d'utilisation et sa personnalisation. Enfin, pour l'accès à la base de données, nous avons intégré Prisma pour sa simplicité et son efficacité dans la gestion des requêtes SQL.

**Environnement de développement intégré** : Nous travaillons principalement avec Visual Studio Code (VS Code) en raison de sa polyvalence, de sa robustesse et de sa large gamme d'extensions, ce qui facilite le développement et la collaboration au sein de l'équipe. Pour les tests unitaires, nous utilisons l'intégration avec GitHub Actions pour automatiser le processus de test et assurer la qualité du code à chaque modification.

**Matériel requis** : Pour les besoins de notre projet, nous utilisons un cluster MySQL hébergé sur Planetscale.com. Cette solution offre une haute disponibilité, une performance optimale et une gestion simplifiée de la base de données, ce qui répond à nos exigences en matière de stockage et de traitement des données. Planetscale ne nous limite pas au nombre d’usager, de lecture ou d’écriture. Il est possible d’utiliser la plateforme gratuitement jusqu’à ce qu’il soit nécessaire d’avoir plus de puissance.

**Interface graphique utilisateur**

Les maquettes sont ici !

Login – Registre

Dashboard home

Cuisines (équipe et création de cuisine)

Livre de Recette

Menu / cout

Fournisseur / ingrédients

Settings

Temps frigo

**Donnée persistante**

Dans notre projet, nous avons choisi d'adopter une approche utilisant deux bases de données distinctes : MySQL et Amazon S3. MySQL sera chargé de conserver l'intégralité des données relatives aux recettes et aux utilisateurs. Quant à Amazon S3, il sera employé pour stocker les médias, tels que les avatars, les images, ou les vidéos liés aux étapes des recettes.

PostgreSQL

Amazon S3

**Structures de données**

Set (rôles dans une cuisine) (coder à la main)

Permet de ne pas avoir de doublons de rôle au sein d’une cuisine

Tuple (nom, rôle)

Ou un custom type, style type User, blablas

Permet de savoir qui tu es et ton rôle lors de l’affichage de certaines pages, ne changera jamais donc tuple ok. De plus, son rôle sera surement lu souvent (ouverture de chaque composante).

Dictionnaire : menu : en JS c’est appeler un objet ? question JC

menu = {

"nom": "Menu Printemps",

"recettes": ["Salade Niçoise" : {

Id : 12hljkh-123nlakjsd-123

"prix": "25.99$"

}, "Poulet rôti", "Tarte aux fraises"],

}

Permet de récupérer facilement le prix pour des stats ou les recettes pour l’affichage

Liste : livre recette(backend)

[id\_recette1, id\_recette2, id\_recette3, id\_recette4]

Type prix [ mesure, ?boite ]

Type boite [qte, unité mesure]

**Patrons de conception**

Façade: A travers une un librairie qui facilite l’utilisation de création de PDF

Data Access Object

Composite : Pour gérer les interactions complexes entre menus, plats, recettes et ingrédients dans votre application, le patron de conception **Composite** pourrait être particulièrement utile. Ce modèle permet de traiter des objets individuels et des compositions d'objets de manière uniforme. Dans votre cas, un menu peut être composé de plats (qui peuvent être vus comme des feuilles ou des composites eux-mêmes), et chaque plat peut être composé d'ingrédients ou de recettes (qui peuvent à leur tour contenir d'autres recettes et ingrédients). Utiliser le modèle Composite faciliterait la gestion de ces structures hiérarchiques et rendrait les opérations sur les structures d'objets plus simples et plus uniformes, par exemple, pour calculer le coût total d'un plat ou d'un menu, ou pour générer une liste d'ingrédients nécessaires.

Observer: Ce patron permettrait de gérer les notifications de changement d'état à divers composants de l'application. Il serait particulièrement utile pour la mise à jour de l'interface utilisateur en réponse aux changements de données, par exemple, la mise à jour des listes de recettes ou des menus.

Stratégie: Conversion des

**Développement d’une bibliothèque**

La bibliothèque permettra de découper d’avantage le visuel.  
micro-biblio :

Conversion des unité de mesure (ml a gramme, qte par boite a kilo)

Création de pdf. Permettre de créer et formatter des pdf à travers une bibliotheque(à travers un design pattern de Facade)

Découpage du visuel react. Permettre de faciliter la creation du visuel a travers une bibliotheque (composante)

**TEXTE :**

Nous prévoyons de développer trois bibliothèques distinctes, chacune ayant pour objectif de renforcer la modularité et la gestion visuelle de notre projet. Voici un aperçu de ces bibliothèques.

1. Bibliothèque de conversion des unités de mesure : Cette bibliothèque fournira des fonctionnalités de conversion des unités de mesure, telles que les millilitres en grammes ou les quantités par boîte en kilogrammes. Bien que cette fonctionnalité soit essentielle pour garantir la cohérence des données, nous mettrons l'accent sur son intégration transparente dans l'interface utilisateur de React, permettant ainsi une présentation claire et intuitive des informations.

2. Bibliothèque de création de PDF : Notre seconde bibliothèque sera dédiée à la création et au formatage de documents PDF. En utilisant un design pattern de Facade, nous simplifierons le processus de génération de PDF tout en veillant à ce que les documents produits soient cohérents avec le reste de l'interface utilisateur de notre application React. Cette approche garantira une expérience utilisateur fluide et homogène.

3. Bibliothèque de découpe du visuel React : Enfin, notre troisième bibliothèque se concentrera spécifiquement sur la découpe du visuel dans React. En fournissant une collection de composants React pré-conçus et modulaires, cette bibliothèque permettra une création rapide et efficace de l'interface utilisateur, tout en assurant une cohérence visuelle à travers l'ensemble du projet. La modularité de ces composants facilitera également la maintenance et l'évolution de notre application au fil du temps.

**Expression régulière**

REGEX  
recherche des ingrédients (création recette)

Javascript filter (regex)  
vérifier adresse Email @, téléphone ok

Question JC, utilisation de la bibliothèque de validation Zod. Qui permet de faire des schéma de de validation de donnée. Typesafe, etc. blabla

**Algorithmes**

Récursivité des ingrédients

Création de recette (récursivité entre recette qui contient n recette)

Ingrédients (fournisseurs)

Recettes (n ing, n rec)

Plats (n rec, n ing)

Menu (n plat)

Suggestion de prix pour les menus en fonction de l’objectif

Algorithme dans la gestion des accès en fonction des rôles des utilisateurs

Algorithme de formatage recette : (commence par majuscule et fini par un point)

**Mathématique**

Conversion unité de mesure

Formattage d’un fichier pdf (interligne, placement de texte)

Calculs couts entre recette / ingrédients et un menu

Moyenne temp frigo ?

**Conception UML**

**ANNEXE**