**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.

Nous avons choisi MySQL car nous pouvons ainsi déployer notre base de données sur *PlanetScale*, qui nous facilite le déploiement du site avec peu de cout au démarrage. Nous avons choisi Amazon S3 afin de simplifier et diminuer le cout des sauvegarder d’image ou de vidéo. De plus, celle-ci utilise des key, value pour sauvegarder les informations.

**Nos Enums**

enum Alergen {

  "lactose"

  "gluten"

  "peanut"

  "nut"

  "egg"

  "fish"

  "crustacean"

  "molluscs"

  "soy"

  "sulphites"

  "mustard"

}

enum ActionMapaq {

  "Plank-Blue"

  "Plank-Red"

  "Plank-Yellow"

  "Cool Quickly"

}

enum UnitMeasure {

  "g"

  "ml"

  "portion"

  "cup"

  "oz"

  "tbsp"

  "tsp"

  "lbs"

}

enum UserType {

  "Admin"

  "Membre"

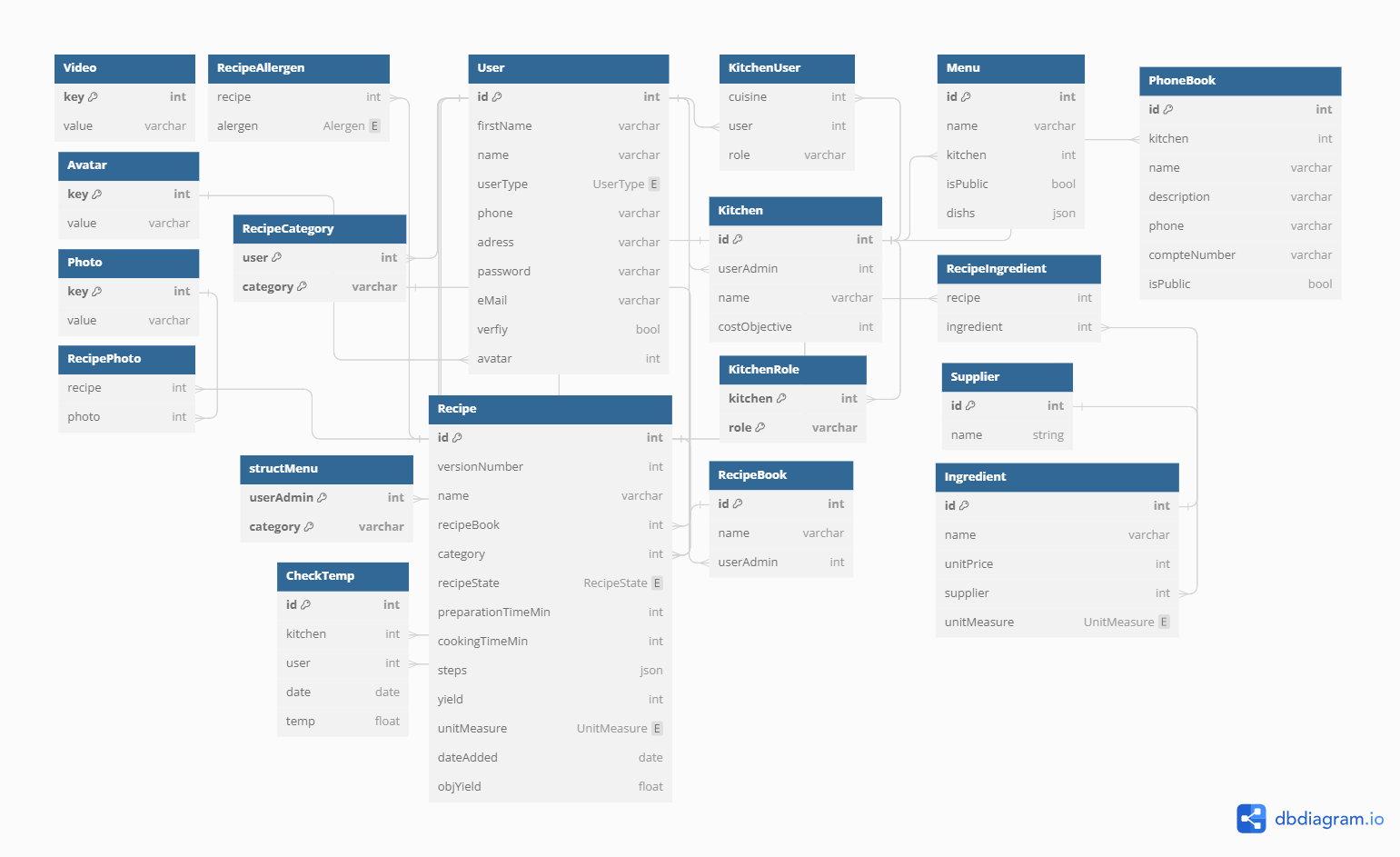
}

enum RecipeState {

  "Recipe"

  "Dish"

}



**Les tables sous Amazon-S3**

* Avatar
* RecipePhoto
* Photo / Vidéo

**Structures de données**

Dans le cadre de notre projet, nous allons utiliser au moins 4 types de structure de donnée.

**Tuple**. Afin de stocker le nom de l’utilisateur et son type de compte. Nous utilisons un tuple, car il n’est pas possible de modifier son type de compte une fois logger dans l’application. Nous voulons aussi rapidement savoir le type afin d’afficher les bonnes composantes react ainsi que les bonnes requêtés au serveur. Readonly.

**Set**. Afin de stocker les différents rôles mis en place au sein d’une cuisine par un chef. Nous utilisons un set, car nous ne voulons pas que les rôles sous dupliqué ou identique et nous n’allons pas indexer sur celui-ci. Nous voulons aussi pouvoir ajouter des rôles.

**Liste**. Afin de différencier les étapes et ingrédients les uns des autres au sein d’une recettes nous allons découper dans des listes les ingrédients distincts les uns des autres. Nous voulons aussi pouvoir concerner l’ordre des étapes.

**Dictionnaire**. Au sein de chacun des menus, nous avons des découpages qui possèdent leur propre valeur. Nous avons un nom, une liste de recettes, une liste de prix ainsi qu’une valeur s il est publique.

menu = {

"nom": "Menu Printemps",

"recettes": ["Salade Niçoise" : {

Id : 12hljkh-123nlakjsd-123

"prix": "25.99$"

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

}

Nous allons programmeur notre propre \_\_\_

**Patrons de conception**

**Façade**. Une Façade est utilisée afin de faciliter la création de PDF au sein de notre projet. Que ce soit pour la création de PDF de recettes, menu ou compte rendu du compte. La façade est utile pour nous simplifier l’interface de création de PDF, un niveau d’abstraction sur la création ainsi que réduire la complexité.

**Data Access Object.** Le DAO nous est pratique pour la communication avec la base de données. Celle-ci regroupera les fonctions, appelables dans le code, qui communiquerons avec la base de données.

**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.

* **Méthodes**
  + Afficher
  + Calculer cost
  + Exporter pdf

**Observer**: Ce patron nous permet 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. De plus, il est pratique pour avoir un retour lors d’un action sur avec un boutton.

**Stratégie**: Le design pattern stategie nous permet lors de l’entrée d’une recette d’avoir le choix sur les unité de mesure(gramme, boite, tasse). Le strategie va découper et appeler le bon algorythme à utiliser pour retourner une unité de mesure en gramme ou mililitre avec la bonne quantité

**Développement d’une bibliothèque**

Dans le cadre de notre projet, nous allons mettre en place trois bibliothèque.

**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. Cette bibliothèque pourrait être pratique dans n’importe quel autre projet qui demanderait des conversion d’unité ou de type d’unité.

**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. Celle-ci pourrait très bien être réutilisable d’un projet à un autre, car finalement elle permet simplement de créer des sections de textes. Elle doit permettre une utilisation simple, mais aussi plus spécifique comme la police d’écriture, la quantité de section possible, le poids des lettres etc.

**Découpage 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**

Nous allons utiliser un REGEX pour vérifier les intrants lors de la création de recettes. Nous voulons vérifier qu’action d’une recette possède au moins 3 mots et que celle-ci commence avec une lettre majuscule. Nous voulons aussi que la phrase finit avec un point. Avec notre regex nous serons en mesure de faire ces vérifications. La majuscule et le point seront automatiquement ajusté par le code et ne refusera pas l’action.

Voici le regex

^[A-Z].\* .\* .\* [a-z].\*\.$

….

Voici nos tests :

…

**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)

La problématique que je souhaite aborder porte sur la gestion optimale des données relatives aux ingrédients, aux recettes et aux plats. Notre objectif est de développer un processus efficace pour la création de recettes qui peuvent inclure à la fois des ingrédients individuels et d'autres recettes, formant ainsi une structure récursive.

Pour répondre à cette problématique, nous avons choisi d'utiliser l'algorithme de la récursivité. Cette décision découle de sa capacité à traiter efficacement les structures imbriquées, telles que les recettes contenant des sous-recettes ou les plats composés d'ingrédients provenant de différentes sources (recette et ingrédients). La récursivité offre une méthode naturelle et élégante pour manipuler ces structures complexes, pour découper menu et recette jusqu’à leur matière primitive.

A travers l'implémentation de cet algorithme il faudra intégrer la gestion des quantités d'ingrédients et prendre en compte les allergies alimentaires de sous recettes. Cependant, la nature modulaire et flexible de la récursivité facilitera ces adaptations.

En résumé, l'algorithme de la récursivité offre une approche puissante et élégante pour résoudre la problématique en lien avec les ingrédients, recettes et menu. La récursivité permettra également de maintenir une structure de données claire et cohérente, facilitant ainsi la gestion et la manipulation des informations culinaires.

**Mathématique**

Dans notre projet, nous allons utiliser certains concepts mathématiques.

**Conversion unité de mesure**. Nous devons être en mesure de récupérer le prix au gramme de certains ingrédients qui seront potentiellement stocker sous forme de quantité par boite.

Nous allons donc devoir utiliser des produits croisés …

**Addition complexe**. Afin de bien se déplacer en (x, y) dans une PDF, nous allons devoir calculer en temps réel lors des insertions à quel niveau en nous y sommes au sein du document. Nous allons devoir additionner la taille de la police (titre, sous-titre, texte) avec le nombre de ligne et la taille des interlignes.

**Calculs de revients**. Cout recette, ingredients, recurcivité pour aller chercher les prix de tous

**Estimation de prix recettes**. Avec l’objectif de gain de la cuisine et les gains de chaque plats.

**Conception UML**

**ANNEXE**