

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 du projet

Nous appuyant sur les fondations établies dans le document de définition, nous visons à développer une solution pratique et adaptable pour les professionnels de la cuisine. Cette application est conçue pour faciliter la formulation des recettes et standardiser les méthodes de travail, tout en assurant un accès facile à l'information pour tous les membres de la brigade. En nous concentrant sur la gestion des coûts, la gestion des ressources humaines adaptée aux brigades de cuisine, et à la conservation des connaissances spécifiques liées aux recettes et aux plats, nous nous engageons à offrir une plateforme qui soutient efficacement la gestion quotidienne. Ce document de conception présente le détail des objectifs et étapes essentielles de la réalisation de ce projet.

Le document de conception a pour objectifs de :

- **Faciliter la maintenance:** offre une référence pour comprendre le système pendant le projet ainsi qu'après le développement.
- **Guider les développeurs:** découpe logique des tâches pour éviter la confusion.
- **Clarification des fonctionnalités:** explique l'approche des problématiques et des caractéristiques spécifiques.
- **Communication efficace:** assure la compréhension unifiée des spécifications et objectifs par toutes les parties prenantes.
- **Réduction des risques:** identifie et définit les défis potentiels en amont.

Infrastructure de développement

Plateforme cible :

Nous développons une application web parce que notre projet vise à être accessible sur toutes les plateformes. Cette approche offre une portabilité intéressante et permet aux utilisateurs d'y accéder sur l'ensemble des appareils disposant d'un navigateur web.

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. Puisque notre application est un projet web, la compréhension de HTML et de CSS est aussi indispensable.

Bibliothèques principales :

Pour le développement de l'interface utilisateur, nous avons choisi d'utiliser React pour sa popularité, sa flexibilité et la richesse de ses fonctionnalités. Nous avons également opté pour Next.js 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 dans le contexte de programmation par composante.

Nous utiliserons aussi la librairie de validation Zod, qui permet de créer des schémas de validation pour les données entrantes et de les valider au fur et à mesure.

Pour les tests nous avons choisi d'utiliser Jest une librairie de test unitaire JavaScript.

Finalement afin de simplifier la création des interfaces graphiques nous avons opté pour la librairie ShadCN UI, celle-ci donne au développeur le code sources des composantes React, nous permettant de se les approprier et de les modifier. Cette librairie est aussi conçue avec Tailwind CSS ce qui renforce le choix d'utiliser cette librairie pour le reste de la mise en page.

Environnement de développement intégré :

Nous travaillons principalement avec Visual Studio Code (VS Code) en raison de sa polyvalence et de sa large gamme d'extensions. Pour renforcer la qualité du code, nous avons intégré les tests unitaires avec GitHub Actions, automatisant ainsi les tests à chaque mise à jour. Nous avons choisi Jest, une bibliothèque de tests JavaScript populaire, pour sa simplicité et l'étendue de documentation disponible.

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.

Matériel requis :

Notre projet s'appuie sur un cluster MySQL hébergé via *Planetscale.com* pour sa fiabilité et sa performance. Cette plateforme s'adapte à nos besoins de gestion de données sans restriction sur le volume d'utilisateurs ou la fréquence des opérations de lecture et d'écriture. *Planetscale* permet un usage initial gratuit, avec la possibilité de passer à des options payantes pour des capacités accrues selon les besoins du projet.

Interface graphique utilisateur

Connexion – Enregistrement

/connexion

The diagram illustrates a user interface for connection and registration. It features a logo at the top left and a main form area. Inside the form, there are two tabs: "Chef" and "Cuisinière". Below the tabs are input fields for "utilisateur" and "mot de passe". At the bottom of the form is a "Si connecter" button. To its right, a link says "pas de compte? S'enregistrer". A large bracket on the right side of the form points to these elements with the following annotations:

- bouton, permet de l'autentification (If OK, redirected to the dashboard).
- lien vers la création de compte.
- input fields
- tabs, permet de changer entre les méthodes de connexion.

/enregistrement (step 1)

The diagram shows a user interface for account creation. It has a title "Créez votre compte chef" and a question "Possédez-vous un compte cuisinière?". Below the question are two buttons: "Oui" and "non". A large bracket on the right side of the interface points to these elements with the following annotation:

↳ passe à step 2

A curved arrow at the bottom points from the "non" button towards the bottom right, with the following annotation:

↳ renvoie à la connexion puis converti le compte en compte chef.

L'enregistrement (step 2)

The diagram shows a registration form with the following fields:

- Logo
- Créer votre compte chd
- num
- prénom
- email
- mdp
- confirmer mdp.
- Accepter les termes et conditions
- Recevoir des info...
- Crée compte

Annotations on the right side of the form:

- input fields
- Radio buttons
- Vérification des champs
- Envoyer email de confirmation
- Crée le compte
- Redirige vers le dashboard

Dashboard

/dashboard (chef)

The dashboard for the chef includes:

- Left sidebar:** Contains icons for C (Chef), R (Recette), \$ (Gestion), and H (Historique). Annotations explain their functions: C closes the menu, R opens the recipe management menu, \$ opens the configuration menu, and H shows the history.
- Header:** Shows the page name "Nom de la page" and a search bar.
- Top right:** Includes a search bar, a help/tutorial button (question mark), and a notification button (info icon).
- Content sections:**
 - Guide de démarrage:** Includes a link "En savoir plus →".
 - Contact favori:** Shows a contact entry with name "Sau 555 5555" and three dots.
 - Nombre de cuisine:** Shows a value of 10.
 - Nombre de recette:** Shows a value of 172.
 - Recette ajoutée récemment:** Shows a recent addition with name "recette" and date "10/02/2024".
 - température frigo:** Shows refrigerator temperatures: frigo 1 at 4°C and frigo 2 at 16°C.
- Bottom right annotation:** Notes that the dashboard allows viewing a summary of important information.

/dashboard (membre)

The dashboard for the member includes:

- Left sidebar:** Contains icons for C (Chef) and H (Historique).
- Header:** Shows the page name "Nom de la page" and a search bar.
- Content sections:**
 - Guide de démarrage:** Includes a link "En savoir plus →".
 - Contact favori:** Shows a contact entry with three dots.
 - Nombre de cuisine dont on est membre:** Shows a value of 12.
 - Dernière menu partagé:** Shows a list of three menu items.

Interface mobile (membre)

/recette (membre)

[mobile]

Nom de la page JC

Poulet au beurre
prep: 20min cook: 40min
rendement: 2500g
alégrine: avan

Ingredients:

--- 2000 g
--- 50 g

Etapes:

1. ---
2. ---
3. ---

/menu (membre)

[mobile]

Nom de la page JC

ENTRÉE:
- - -
- - -
- - -

PLATS:
- - -
- - -
- - -

/bottin (membre)

[mobile]

Nom de la page JC

Beigade:
- - -
- - -
- - -

Végences:
- - -
- - -
- - -

/ajout - température

[mobile]

Nom de la page JC

Reiso 1 ▾

4 °C ▾

ajouter

/dashboard (membre)

[mobile]

Nom de la page JC

ouvre le menu d'option

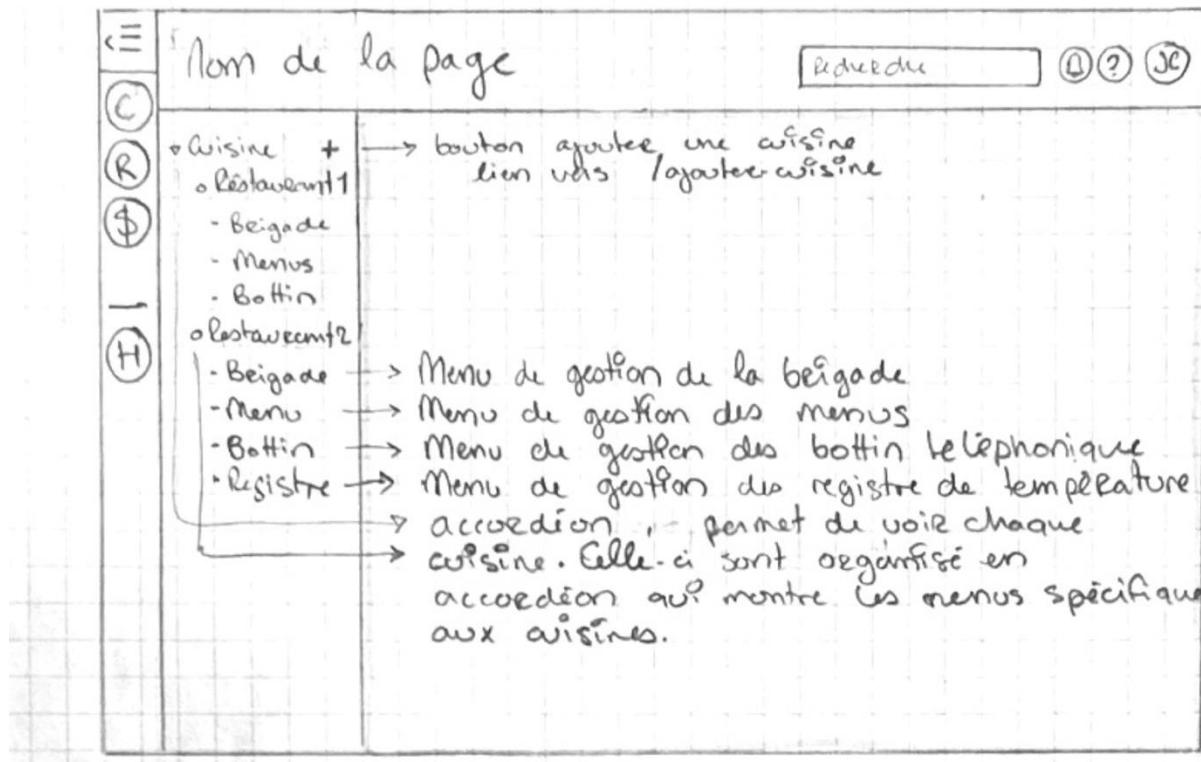
Guide du déneige
En Savoie plus →

Menus préfér.

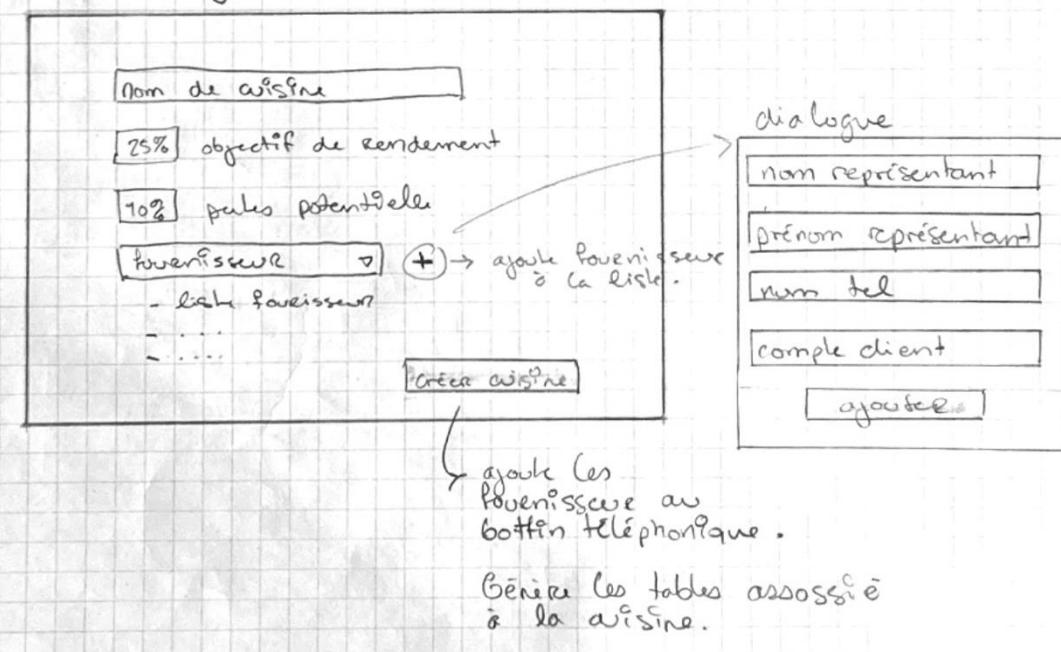
Contact favorit

Cuisine

/cuisine

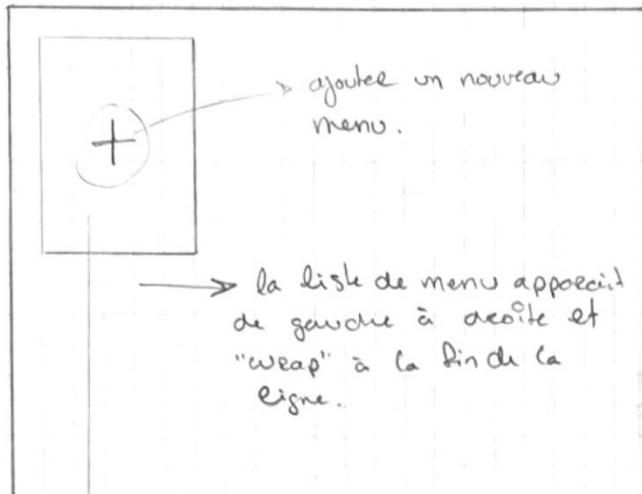


/cuisine/ajoutee_cuisine



Menu

/wɪsɪn/ /id.-wɪsɪn/ gestion-menu



Dialogue

- nom du menu
- date picker
- privacy
- ajoutee

/wísine /id-wísine /gestion-menu /id-menu

permet de l'organiser sur la page des sections

Catégorie + → ajoute une catégorie à la page

☰ ENTRÉE - → retire catégorie

nom du plat	cost	objectif	prix vente
Spas bolo	4,90	25 %	22,50

ajouter un plat + → ajoute un plat à la section

☰ PLAT PRINCIPAUX - → recherche dynamique dans la banque de plats.

Saviez-vous que...

→ peut-être modifiable.

→ initialement pourpre avec l'objectif de base mis en place dans la cuve.

→ com posante style
table coulenant les
plats ajouté

→ Sauvegarde le menu
Redirige à la page précédente

Bottin

/wɪzɪn /id-wɪzɪn / bottin

nom, prénom	adresse/entreprise	tel	client
Chvet, Rémi	Restaurant 1	555-5555	
Turgeon, Jean	Hector Léveillé	555-5555	M645

ajouter
au
favoris

ajouter contact

Ajouter un contact

nom

prénom

entreprise

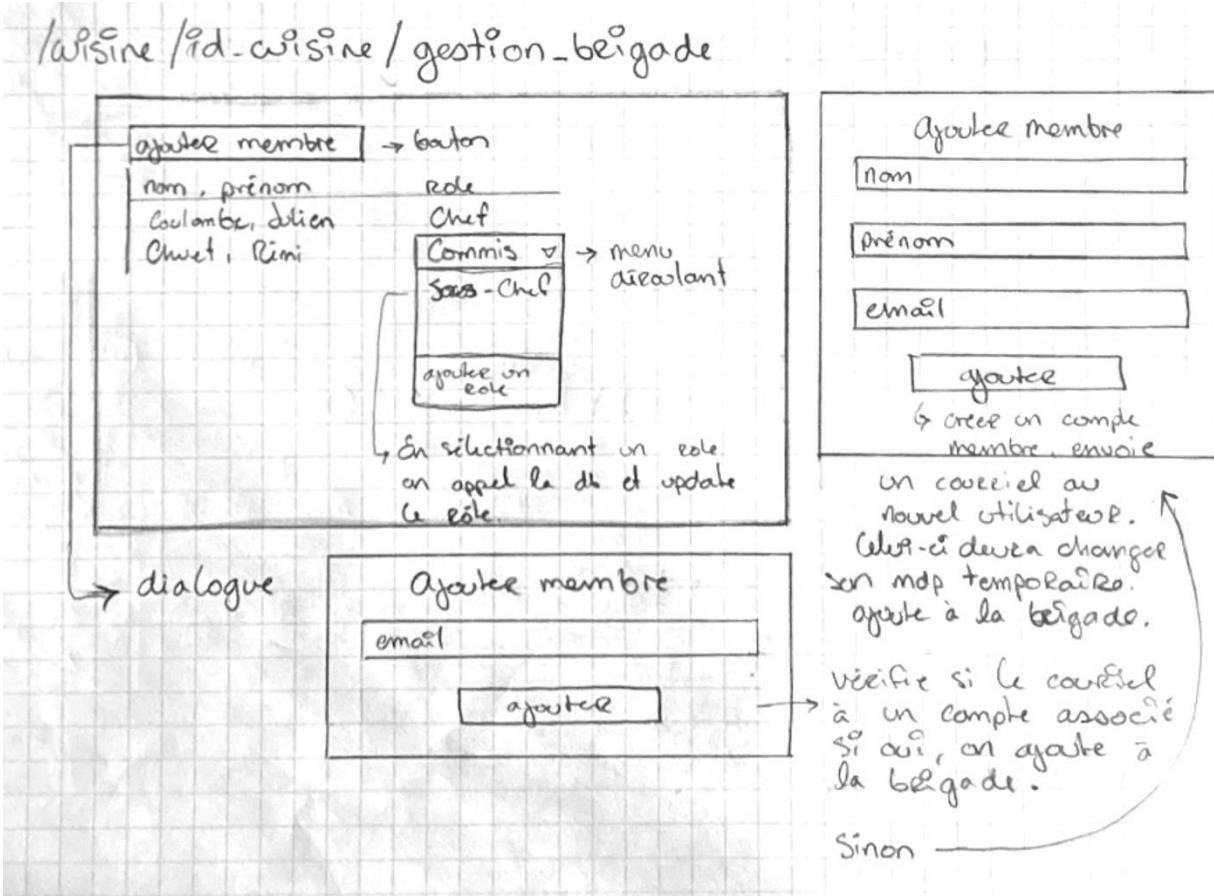
telephone

compte client

ajouter

→ ajoute un contact à
la liste de contact.

Brigade



Registre température

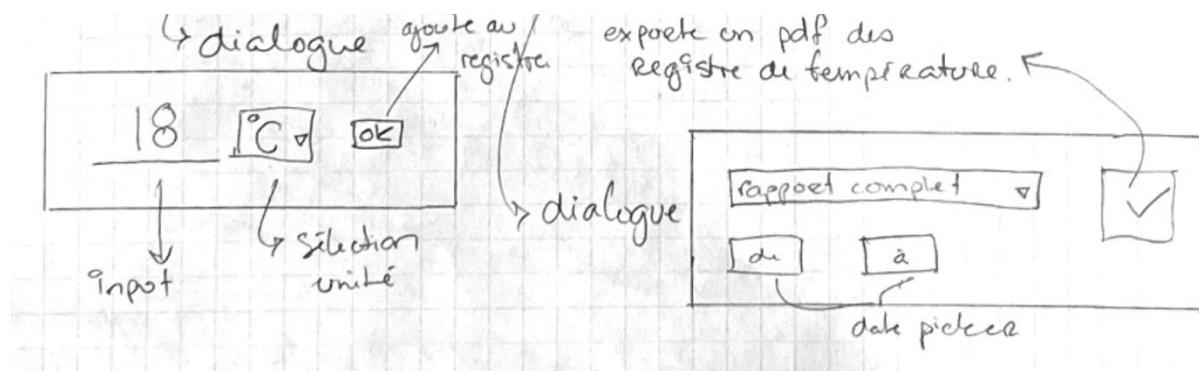
/wísine /id-wísine /registree

feigo 1	+	ajouter un feigo	
date	time	température	employé
22-02-2024	14:00:00	18°C	Clém, Lémi

→ modifiable

[ajouter une temp]

[exportee]



Recettes

/recette

☰	Nom de la page	Recherche.	☰	?	☰
C	↳ Cire recette + o Default - catégories - catégories	→ groupée en Cire de recette. → peut être renommé comme un dossier			
R	o Fissant - catégories	Bouton qui mène au Cire de recette			
\$		→ on peut aussi filtre directement en catégorie			
H					

Livres de recettes

/recette / id-Cuve /

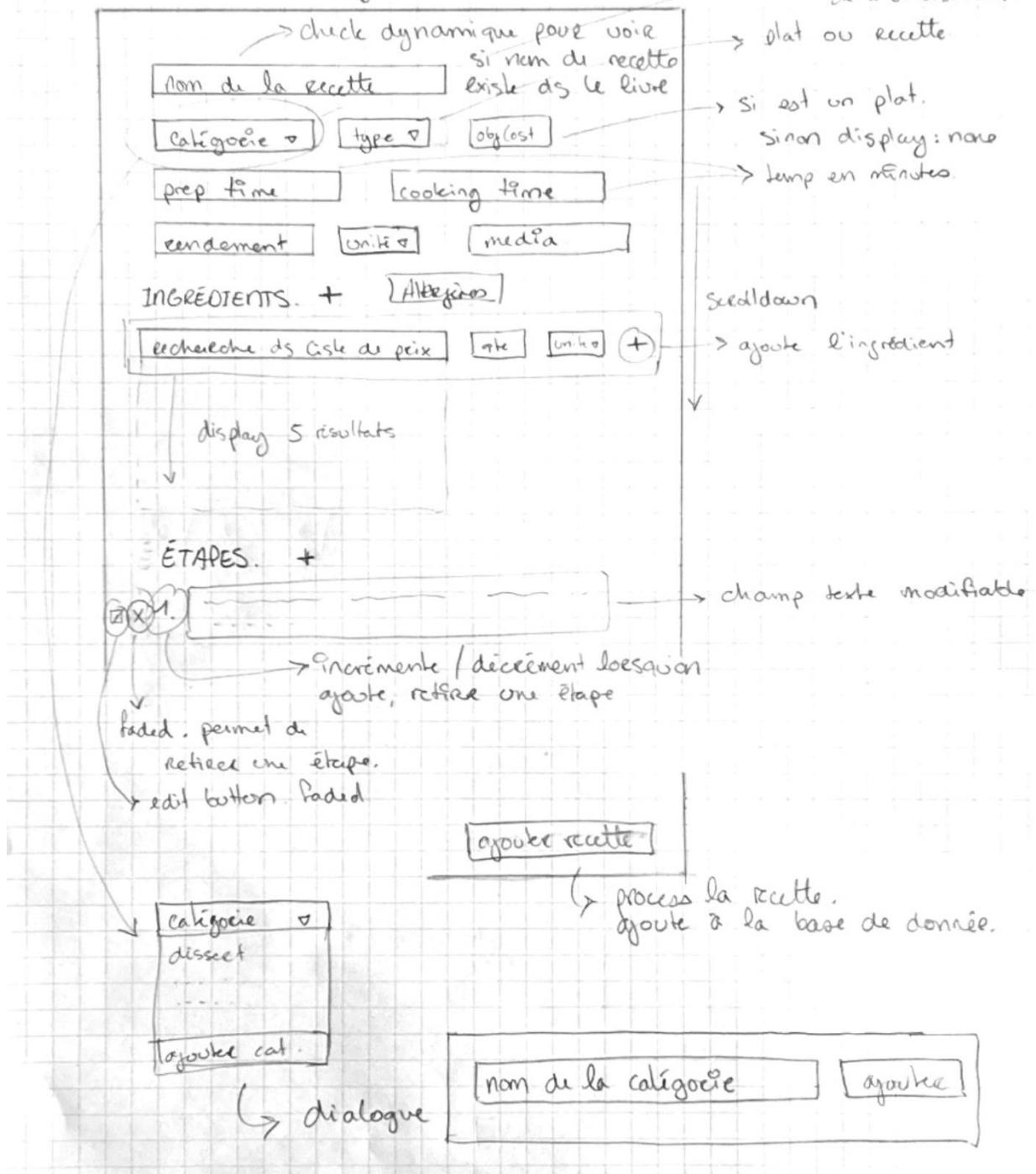
recette
incomplie.

nom recette	catégorie	rendement	cost	date
vinaigrette ail	vinaigrette	1000ml	6,50	02-24-24
① ail	condiment	1000 ml	8,48	03-21-24
...
回 1, 2, ..., 5 回				
ajouter recette				

/recette /ajouter_Cuve_recette

nom du livre de recette
Cuve Cuvé

/recette /Id_livre /ajoutee_recette



Liste de prix

/liste-de-prix /id-bouenisseur

<input type="checkbox"/>	Nom de la page	recherche	<input type="radio"/> <input type="radio"/> <input type="radio"/> JC								
<input checked="" type="radio"/> C <input type="radio"/> R <input type="radio"/> \$ <hr/> <input type="radio"/> H	<ul style="list-style-type: none"> • bouenisseur • Maechi • Hector L. • bleu Marin • La mée • Canam Maechi	02-22-2021	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>code</th> <th>nom</th> <th>prix</th> <th>qte</th> </tr> </thead> <tbody> <tr> <td>01234</td> <td>carottes "coenpact"</td> <td>22,99\$</td> <td>40 lbs</td> </tr> </tbody> </table> <p>[ajouter ingrédient]</p>	code	nom	prix	qte	01234	carottes "coenpact"	22,99\$	40 lbs
code	nom	prix	qte								
01234	carottes "coenpact"	22,99\$	40 lbs								

derrière
⇒ date de
mise à jour

→ seulement dispo
dans le maechi.

Sincr

[update pricecliste]

↳ prezee CSV?

Settings

/ user-setting

	Nom de la page	
 — 	<p>Profil</p> <p>email</p> <p>mot de passe</p>	<p>img. <input type="text"/> ajouter</p> <p><input type="text"/> nom telephone</p> <p><input type="text"/> date père → date naissance</p> <p><input type="button" value="modifier"/> ajouter ↳ info perso</p> <p><input type="button" value="modifier"/> changer le mdp.</p>

✓ ne peut pas être édité manuellement.

Donnée persistante

Nous avons opté pour une structure de données sur deux niveaux. Pour la gestion des informations telles que les recettes et les profils utilisateurs, nous utilisons MySQL en combinaison avec *PlanetScale* pour une mise en place efficace et économique au départ.

Parallèlement, Amazon S3 est notre choix pour le stockage des médias associés aux recettes, comme les images et vidéos, grâce à son système de clés et valeurs qui facilite la gestion et réduit les coûts de sauvegarde.

Nos énumérations :

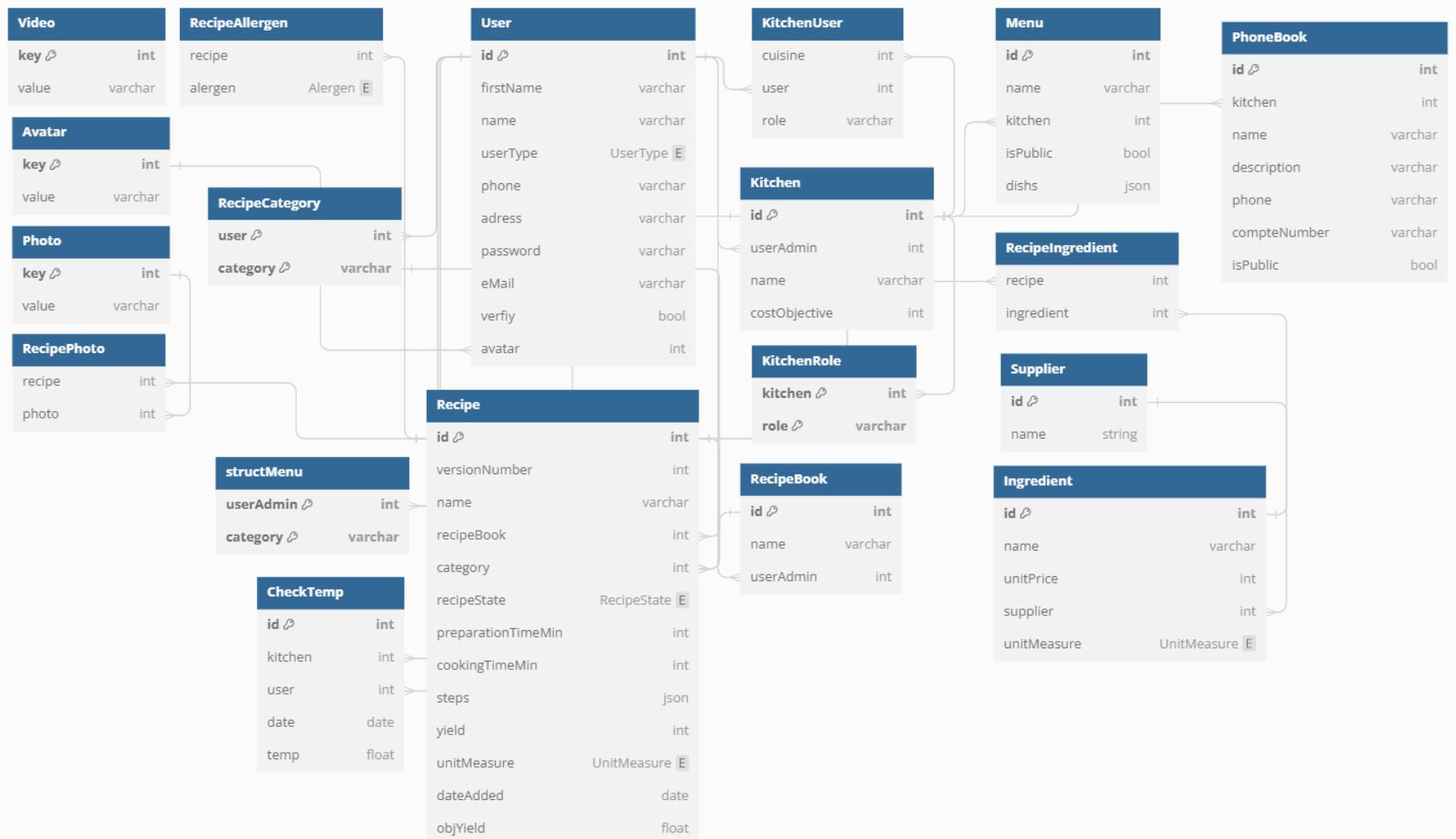
```
enum UserType {  
    "Admin"  
    "Membre"  
}
```

```
enum RecipeState {  
    "Recipe"  
    "Dish"  
}
```

```
enum UnitMeasure {  
    "g"  
    "ml"  
    "portion"  
    "cup"  
    "oz"  
    "tbsp"  
    "tsp"  
    "lbs"  
}
```

```
enum Alergen {  
    "lactose"  
    "gluten"  
    "peanut"  
    "nut"  
    "egg"  
    "fish"  
    "crustacean"  
    "molluscs"  
    "soy"  
    "sulphites"  
    "mustard"  
}
```

```
enum ActionMapaq {  
    "Plank-Blue"  
    "Plank-Red"  
    "Plank-Yellow"  
    "Plank-Green"  
}
```



Les tables sous Amazon-S3

- Avatar
- Photo
- Vidéo

Structure de données

Dans le cadre de notre projet, nous utilisons au moins quatre (4) types de structure de données. Le tuple, le set, la liste et les objets (dictionnaire javascript).

Arbre:

L'arbre est une structure de données optimale pour notre projet, car il épouse parfaitement la complexité et la nature hiérarchique de la gestion des recettes de cuisine. Cette structure nous permet de représenter de manière intuitive et efficace des recettes, qui elles-mêmes se décomposent en sous-recettes et ingrédients. La force de l'arbre réside dans sa capacité à traiter chaque composant avec la même interface. Nous allons mettre en œuvre par nous-mêmes cette structure de données au sein de notre projet.

Tuple :

En TypeScript, il est possible de définir un tuple pour contenir le nom d'utilisateur et son type de compte. Cette structure est utilisée parce que le type de compte ne peut pas être modifié après la connexion. L'utilisation d'un tuple *readonly* permet de garantir l'immutabilité et d'accéder rapidement aux informations nécessaires pour afficher les composants React appropriés et effectuer les bonnes requêtes au serveur.

Set :

Pour gérer les rôles dans une cuisine, les catégories de recettes et les menus, un *set* est utilisé. Ce choix permet d'éviter les doublons et garantit l'unicité des éléments. De plus, comme l'indexation n'est pas requise et l'ajout de nouveaux rôles est nécessaire, le *set* se révèle être un outil adapté à ces besoins.

Liste :

Pour garder une organisation claire des menus et des livres de recettes, les arbres et les étapes en listes distinctes. Cette méthode nous permet de différencier nettement chaque recette et de suivre l'ordre des étapes de préparation. L'utilisation de listes est appropriée, car elle n'exige pas la nomination de clés spécifiques, ce qui simplifie la structure de nos données. Dans ces cas la complexité n'est pas vraiment importante puisque nous chercherons à extraire chaque valeurs de la liste sans avoir besoin de chercher au travers de celle-ci.

Objets (Dictionnaire en Javascript) :

Dans notre projet, nous utilisons des objets JavaScript, qui sont similaires aux dictionnaires en Python, pour stocker et manipuler les données. Ces objets seront utiles pour conserver des informations structurées, comme les coordonnées des représentants. Ils permettent une organisation claire et l'accès rapide aux données. L'utilisation des objets JavaScript nous permet aussi de passer de manière simple des propriétés à tous les composants React.

```
représentantLegume = {  
    "nom": "Jean Turgeons",  
    "telephone": "514-555-5555",  
    "entreprise" : "Hector Larivé ",  
    "compteClient" : "C1234"  
}
```

Cette utilisation de la notation d'objet Javascript s'étendra à plusieurs autres systèmes de notre application. Essentiellement lorsque nous aurons besoin d'accéder à certaines données directement sans vouloir itérer sur une liste.

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 ou des menus. 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 d'utilisation. Nous avons consulté un article de [GeeksForGeeks](#) afin de mieux comprendre comment le patron de conception se met en place.

Data Access Object :

Le DAO nous est pratique pour la communication avec la base de données. Celle-ci regroupe les fonctions, appelables dans le code, qui communiqueront avec la base de données. Notre application repose sur la communication entre plusieurs systèmes qui nécessite chacun des accès précis à certains types de données. Nous avons donc choisi de séparer nos DAO en plusieurs objets spécifiques qui nous permettent d'encapsuler la logique d'accès aux données pour chacun de ces systèmes.

Observer:

Ce patron nous permet de gérer les notifications de changements d'état qui s'affiche via les 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'une action à partir d'un bouton.

Stratégie:

Le patron de conception stratégie, utilisé dans notre système de conversion d'unités pour les ingrédients de recettes, permet une grande flexibilité en offrant plusieurs algorithmes de conversion adaptés à différentes unités de mesure (grammes, boîtes, tasses, etc.). Cette approche encapsule chaque algorithme de conversion, les rendant interchangeables et indépendants des clients qui les utilisent.

Ainsi, lors de l'entrée d'une recette, le système identifie l'unité de mesure spécifiée pour chaque ingrédient et sélectionne automatiquement l'algorithme de conversion approprié. Cela assure une conversion précise des quantités en unités standards (grammes pour les solides, millilitres pour les liquides), quelle que soit l'unité d'origine. Ainsi, le modèle stratégique simplifie le processus de conversion.

Composite :

Notre projet nécessite la gestion d'une structure complexe de données, où les menus se composent de plats, qui à leur tour se composent de recettes et d'ingrédients, avec la particularité que les recettes puissent elles-mêmes contenir d'autres recettes ou ingrédients.

Le défi principal consiste en ce que la manipulation de cette hiérarchie se fasse de manière homogène, en permettant des opérations cohérentes sur l'ensemble des éléments, qu'ils soient composites ou primitifs. Le patron de conception Composite offre une solution élégante en nous permettant de traiter chaque élément de la hiérarchie de manière uniforme.

Implémentation :

Dans le cadre de notre projet, nous mettrons en œuvre de manière autonome plusieurs patrons de conception, sélectionnés, car ils répondent aux objectifs spécifiques de notre application.

Nous utiliserons la Façade pour simplifier les interactions avec la librairie de génération de documents PDF, le Composite pour construire une hiérarchie d'objets menus-recettes-ingrédients, le DAO pour encapsuler l'accès aux données, et la Stratégie pour permettre une flexibilité lors de la conversion des unités de mesure.

Nous avons choisi de développer entièrement le Composite. Ce choix est motivé par la structure de notre application qui gère une combinaison d'objets avec des propriétés récursives, allant des menus aux ingrédients primitifs. L'implémentation du Composite nous permettra de traiter ces objets de manière uniforme, facilitant ainsi le développement et la maintenance du code.

Nous prévoyons encapsuler plusieurs comportements tels que le calcul des coûts de revient, l'affichage des informations pertinentes, comme le nom, la quantité, les allergènes des menus, plats, recettes ou des ingrédients. Certains de ces objets, plus complexes, comme les recettes auront des comportements distinctifs comme l'affichage de la méthodologie. Les actions d'insertion et de suppression seront aussi gérées par la structure de données résultante.

Pour guider notre développement, nous consulterons des ressources telles que le site [Refactoring Guru](#), la série *Youtube* sur les patrons de conception par [Derek Banas](#) ainsi que toutes autres ressources qui seront jugées nécessaires lors de l'implémentation.

Développement d'une bibliothèque

Dans le cadre de notre projet, nous allons mettre en place deux bibliothèques concrètes.

Conversion des unités de mesure :

Sélection du module technique réutilisable:

L'analyse du projet révèle que la conversion d'unités de mesure est un processus fondamental à la gestion de cuisine, mais aussi extensible à d'autres domaines. Ce besoin justifie la création d'une bibliothèque dédiée, capable de gérer une variété d'unités et de conversions, ce qui la rend utile dans plusieurs types de scénarios.

Justification:

La sélection de ce module pour un développement en tant que bibliothèque universelle repose sur son applicabilité étendue. Par exemple, dans un projet culinaire, convertir des tasses en grammes peut être essentiel pour la précision des recettes. La capacité de cette bibliothèque de s'adapter à divers besoins et contextes souligne son utilité et son potentiel de réutilisation.

Implémentation :

Pour faciliter le développement, l'implémentation de la bibliothèque s'appuie sur le patron de conception stratégie. Cela permet de décomposer le processus de conversion en plusieurs stratégies spécifiques, facilement interchangeables. Chaque stratégie est responsable d'un ensemble d'unités de mesure ou d'un type de conversion particulier, rendant le système flexible et facile à étendre avec de nouvelles unités ou méthodes de conversion. La bibliothèque est développée en veillant à l'encapsulation, à la modularité, et à l'indépendance du code, facilitant ainsi son intégration dans d'autres projets.

Génération de PDF :

Sélection du module technique réutilisable:

L'une des difficultés relevées dans l'analyse de notre projet consiste à exporter les recettes en format PDF formaté de manière précise et esthétique. Pour ce faire, il existe déjà quelques librairies qui effectuent ce travail, cependant celles-ci sont généralement difficiles à utiliser et à configurer.

Justification:

Pour remédier à cela, notre bibliothèque a pour objectif de simplifier le processus de création de PDF. Elle agit comme une surcouche à la librairie existante PDFME, offrant une expérience utilisateur simplifiée et accessible. Les utilisateurs pourront ainsi créer des objets personnalisés de configuration, permettant de définir le formatage des documents, tels que la police, le poids du texte, le nombre de sections, et bien plus, de manière intuitive et répétable.

Implémentation :

L'implémentation de cette bibliothèque se fait à travers le patron de conception façade, qui offre une interface de programmation simplifiée masquant la complexité sous-jacente des opérations de génération de PDF. En encapsulant les interactions avec la librairie sous-jacente, nous garantissons une intégration fluide et sans heurt, offrant ainsi une abstraction efficace pour les développeurs. Cette approche permet une intégration transparente au sein de notre application React, assurant une cohérence visuelle et fonctionnelle avec l'interface utilisateur existante.

Ultimement notre projet permet de créer plusieurs autres bibliothèques qui pourraient être utiles pour des projets internes futurs. Par exemple la création d'une bibliothèque de composants React.

Expression régulière

Dans notre système, nous avons intégré l'utilisation d'expressions régulières pour valider les mots de passe. L'expression régulière conçue impose que chaque mot de passe comporte un minimum de 8 caractères et inclut au moins une lettre majuscule et un chiffre. Bien que l'utilisation de caractères spéciaux ne soit pas obligatoire, elle est permise.

`^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$` ¹

Correct

The screenshot shows a regular expression testing interface. The regular expression input field contains `^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$`. The test string input field contains `P@s!sw0R23rwAd`. The results panel indicates "1 match (0.2ms)" with a green button labeled "/ gm". The match information shows "Match 1 0-14 P@s!sw0R23rwAd".

Correct

The screenshot shows a regular expression testing interface. The regular expression input field contains `^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$`. The test string input field contains `Passwor@d1!23`. The results panel indicates "1 match (0.0ms)" with a green button labeled "/ gm". The match information shows "Match 1 0-13 Passwor@d1!23".

Le mot de passe ne possède pas de lettre majuscule

The screenshot shows a regular expression testing interface. The regular expression input field contains `^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$`. The test string input field contains `password`. The results panel indicates "no match (0.0ms)" with a grey button labeled "/ gm". The match information panel displays the error message "Your regular expression does not match the subject string."

¹ Source: Password Validation REGEX (02-03-2024), <https://regextester.com/3bf81>

Le mot de passe n'est pas assez grand

The screenshot shows a regular expression tester interface. On the left, under "REGULAR EXPRESSION", is the pattern `^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[a-zA-Z]).{8,}$` with the option "/gm" selected. The status bar indicates "no match (0.0ms)". On the right, under "EXPLANATION", it says "Your regular expression does not match the subject string." Below this, under "MATCH INFORMATION", is the same message. In the "TEST STRING" input field, the user has entered "passoR0".

Le mot de passe ne possède pas de chiffre

This screenshot is similar to the previous one, showing the same regular expression and test string. The difference is in the "TEST STRING" field, where the user has now entered "passeroR|". The "EXPLANATION" and "MATCH INFORMATION" sections still state that the regular expression does not match the subject string.

Algorithmes

Calculs des coûts de revient :

Le calcul précis des coûts de revient d'un plat est essentiel pour la gestion des prix et la rentabilité d'un restaurant. La complexité réside dans l'agrégation des coûts des ingrédients individuels et des sous-recettes (celle-ci ayant une nature récursive).

À la suite d'une discussion avec Pierre-Paul Monty (professeur au CVM) l'algorithme de parcours infixé ([In-Order Traversal](#)) a été choisi pour sa capacité à traiter les structures d'arbres. Dans notre application les nœuds représentent des recettes individuelles et les feuilles des ingrédients ou des ensembles de sous-recettes.

Le choix de cet algorithme est pertinent, car il permet de visiter tous les composants d'une recette en respectant l'ordre nécessaire à leur combinaison, nous permettant de faire le calcul des coûts, mais aussi de retourner des informations telles que les quantités, ou encore les allergènes maintenus dans les recettes. La complexité de l'algorithme est de $O(n + m)$, où n est le nombre d'ingrédients et m le nombre de sous-recettes.

Bien que l'implémentation standard d'un parcours infixé puisse être utilisée, elle nécessite une adaptation pour gérer correctement les coûts cumulatifs des sous-recettes. Cette adaptation comprend l'introduction de variables supplémentaires pour le cumul des coûts et une logique de traitement différente pour les ingrédients et les sous-recettes. Lorsqu'une sous-recette sera calculée, nous devrons faire parvenir à la méthode de calcul la quantité utilisée par la racine. Nous pourrons alors calculer le ratio entre le prix au rendement de la recette et la quantité requise pour la recette.

En utilisant le parcours infixé pour faire le traitement de cette information, nous appliquons un algorithme éprouvé et efficace, adapté au type de structure utilisé, tout en tenant compte de la spécificité des données de notre domaine.

Mathématique

Notre application vise à simplifier la gestion de recettes de cuisine, qui comprend le calcul des coûts des ingrédients, l'estimation du prix des recettes, et la génération de documents PDF pour les utilisateurs. Le défi consiste à intégrer précisément les mathématiques pour réaliser ces tâches avec exactitude et efficacité.

Nous appliquerons les concepts mathématiques suivants :

Conversion des unités de mesure :

Notre application effectue la conversion des mesures solides (telles que livres, kilogrammes, onces) en grammes et celle des liquides en millilitres. Cette étape est cruciale pour assurer la cohérence des recettes, indépendamment des systèmes de mesure utilisés par nos utilisateurs. Nous employons des produits croisés pour transformer ces unités de mesure et pour recalculer le prix des ingrédients selon leur nouvelle unité. Ce processus de conversion est essentiel pour maintenir une gestion précise et uniforme des coûts.

Nous reconnaissons que cette approche présente certaines limitations, notamment en ce qui concerne la précision des conversions de prix. Les prix convertis peuvent résulter en des valeurs très petites. Afin de conserver une précision raisonnable, la base de données enregistre les valeurs monétaires en centimes sous forme d'entiers. Cela nous permet de gérer les prix avec une plus grande précision, tout en évitant les problèmes potentiels de représentation de nombres flottants, qui peuvent introduire des erreurs d'arrondi supplémentaires.

Vecteurs :

Bien que la génération de PDF soit un aspect secondaire de notre application, la mise en page des documents sera gérée par des vecteurs en coordonnées (x, y), ajustant la position du texte selon la taille de la police, le nombre de lignes et la hauteur des interlignes, pour assurer un rendu professionnel et lisible des recettes.

Récursivité :

Pour déterminer avec précision le coût de revient et estimer les prix de vente des plats, notre application fait appel à des fonctions récursives. Cette approche algorithmique est idéalement adaptée pour naviguer à travers la structure composite des recettes qui peuvent elles-mêmes comprendre des ingrédients et d'autres recettes. La récursivité nous permettra d'accumuler les coûts à chaque niveau de cette structure, en ajoutant les coûts des ingrédients de base et en multipliant par les quantités nécessaires pour chaque recette.

Bien que les opérations réalisées à chaque appel récursif soient simples — essentiellement des additions et des multiplications — la puissance de cette méthode réside dans sa capacité à traiter des structures

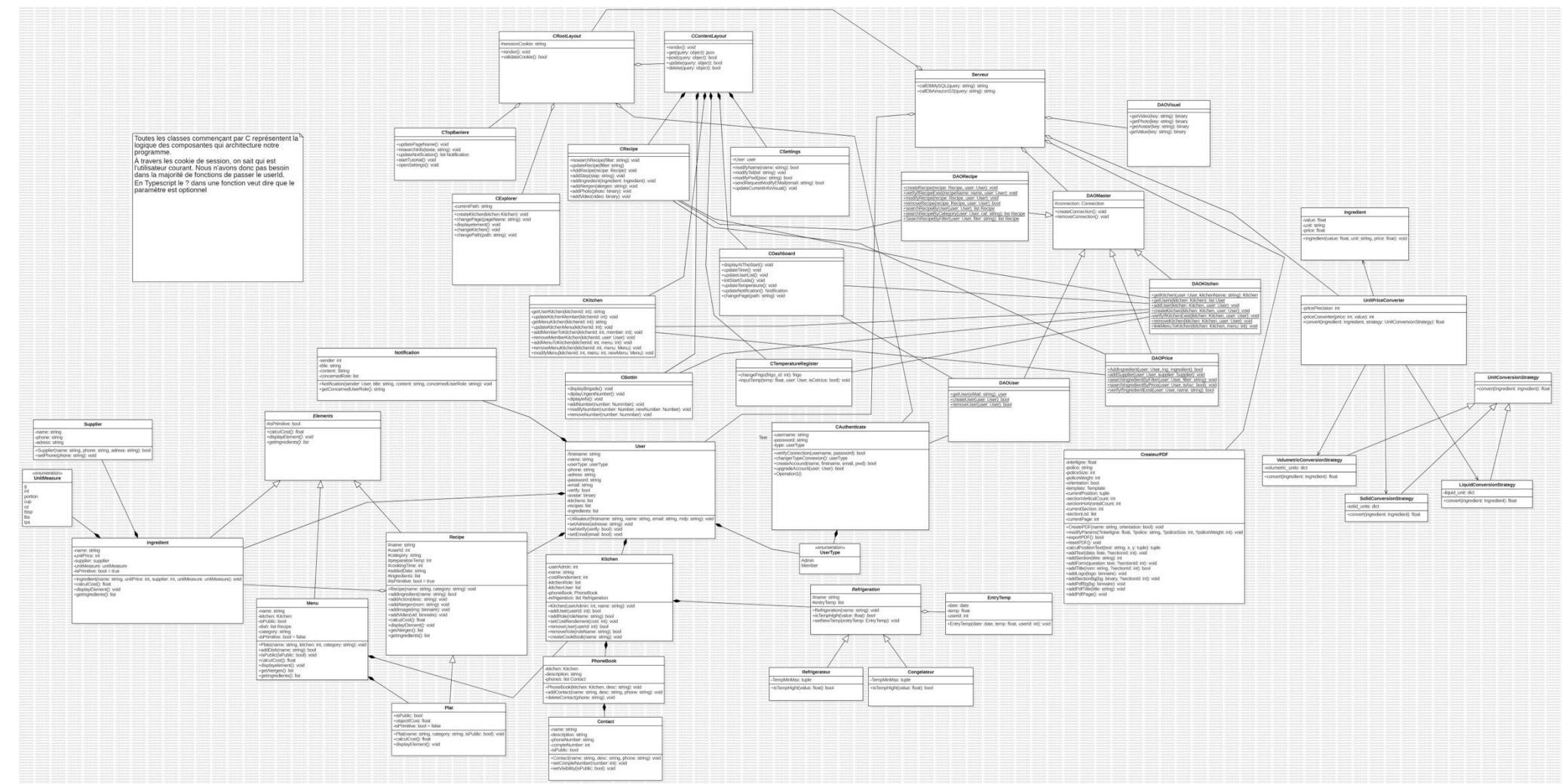
complexes de manière élégante et maintenable. Néanmoins, nous sommes conscients des risques associés à la récursivité, notamment le risque de débordement de pile dû à des appels récursifs excessifs.

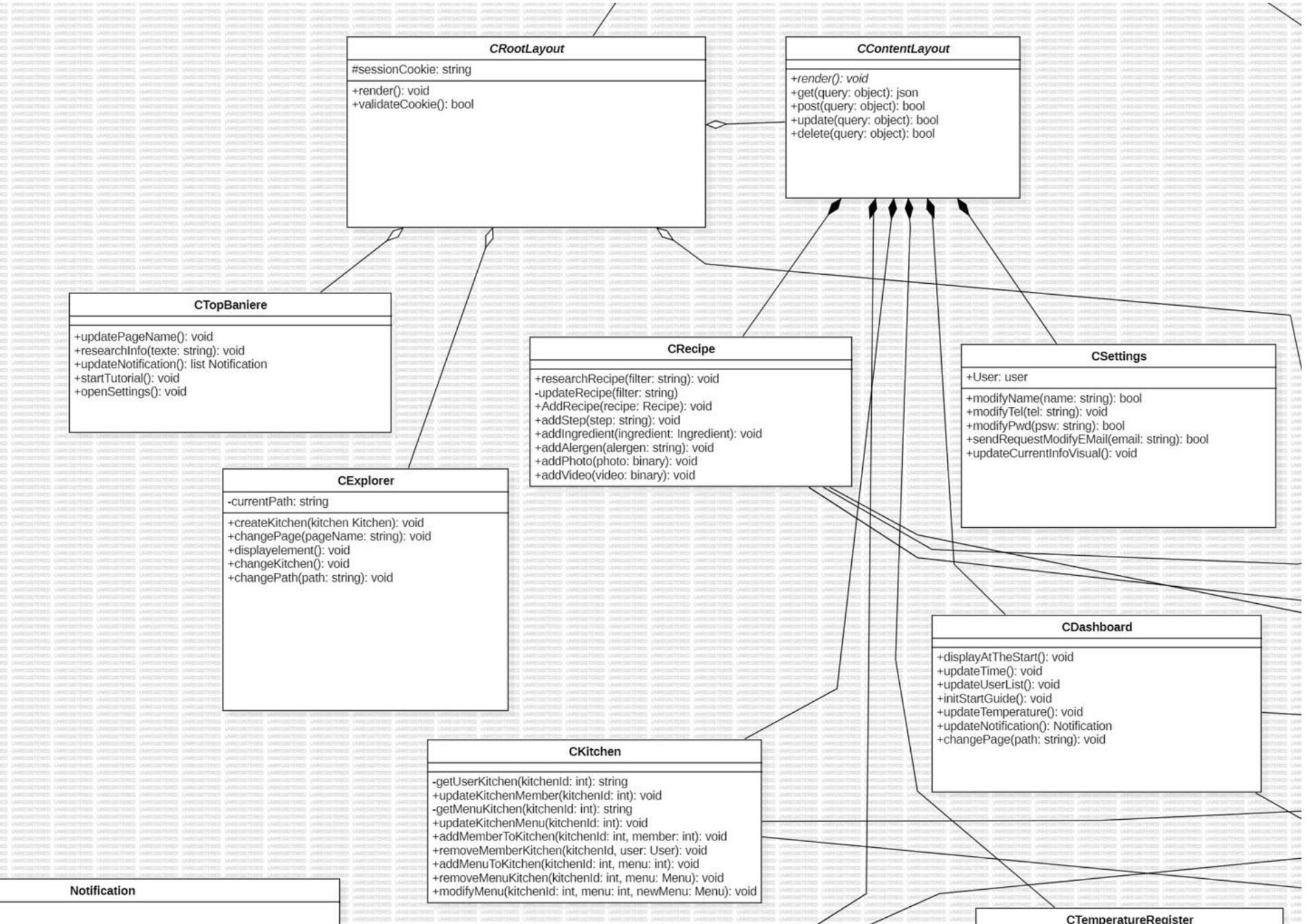
Pour prévenir de tels problèmes, nous mettons en œuvre des cas de base bien définis pour assurer que chaque appel récursif atteigne un point d'arrêt. De plus, nous optimisons nos structures de données et révisons notre logique récursive pour minimiser la profondeur des appels récursifs. En prenant ces précautions, nous pouvons exploiter les avantages de la récursivité tout en atténuant les risques.

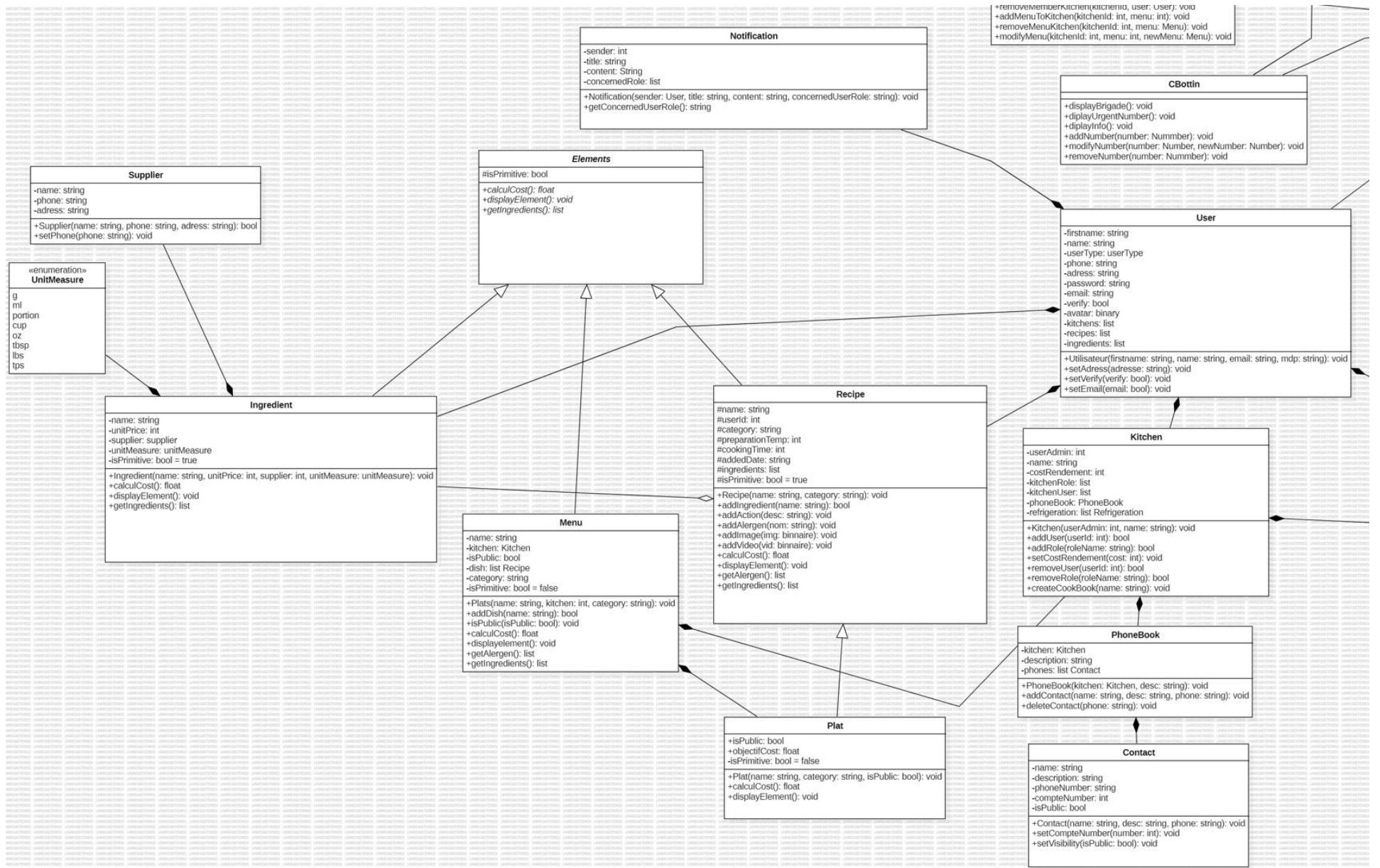
Conception UML

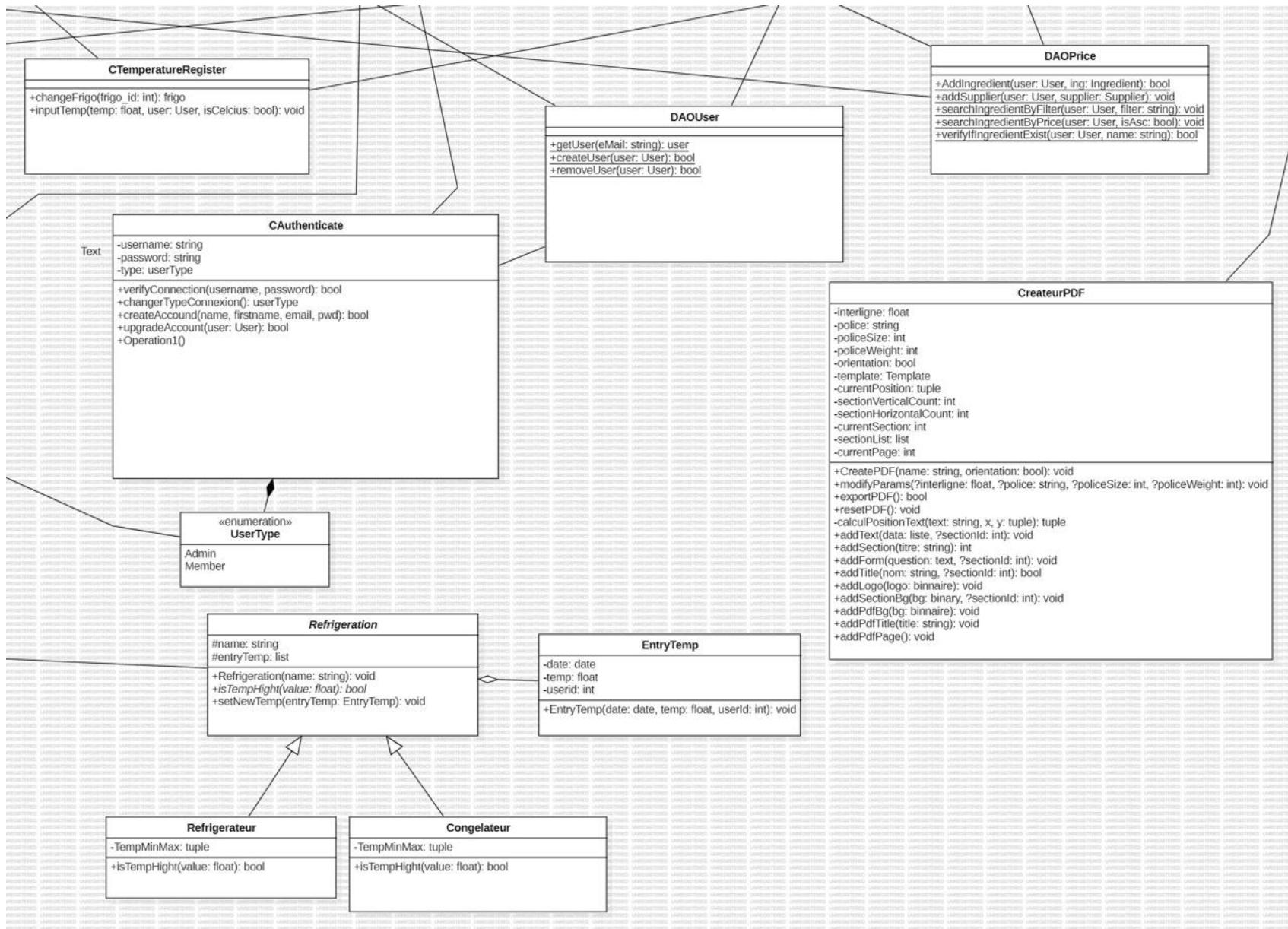
Diagramme de classes :

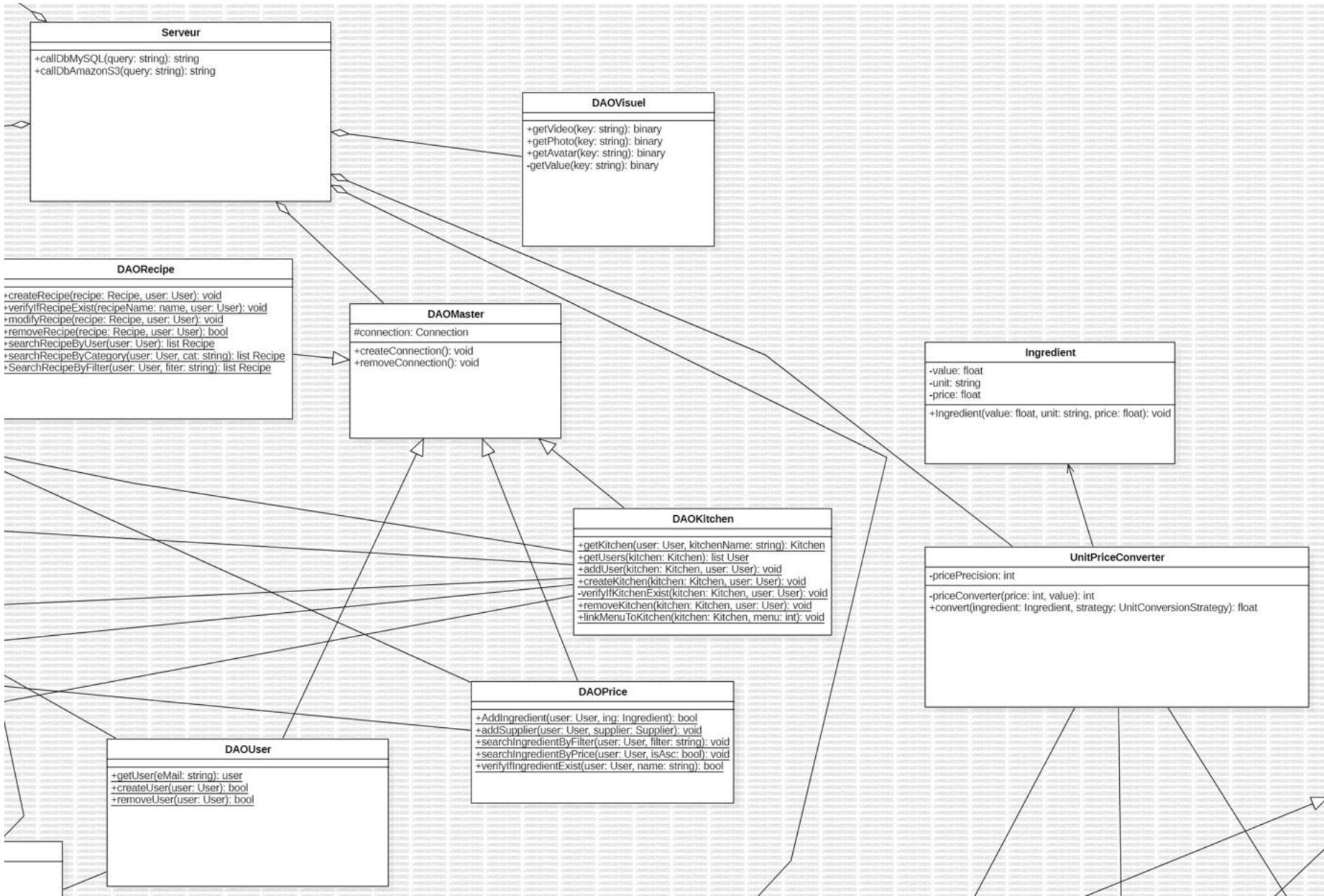
Toutes les classes commençant par C représentent la logique des composantes qui architecture notre programme.

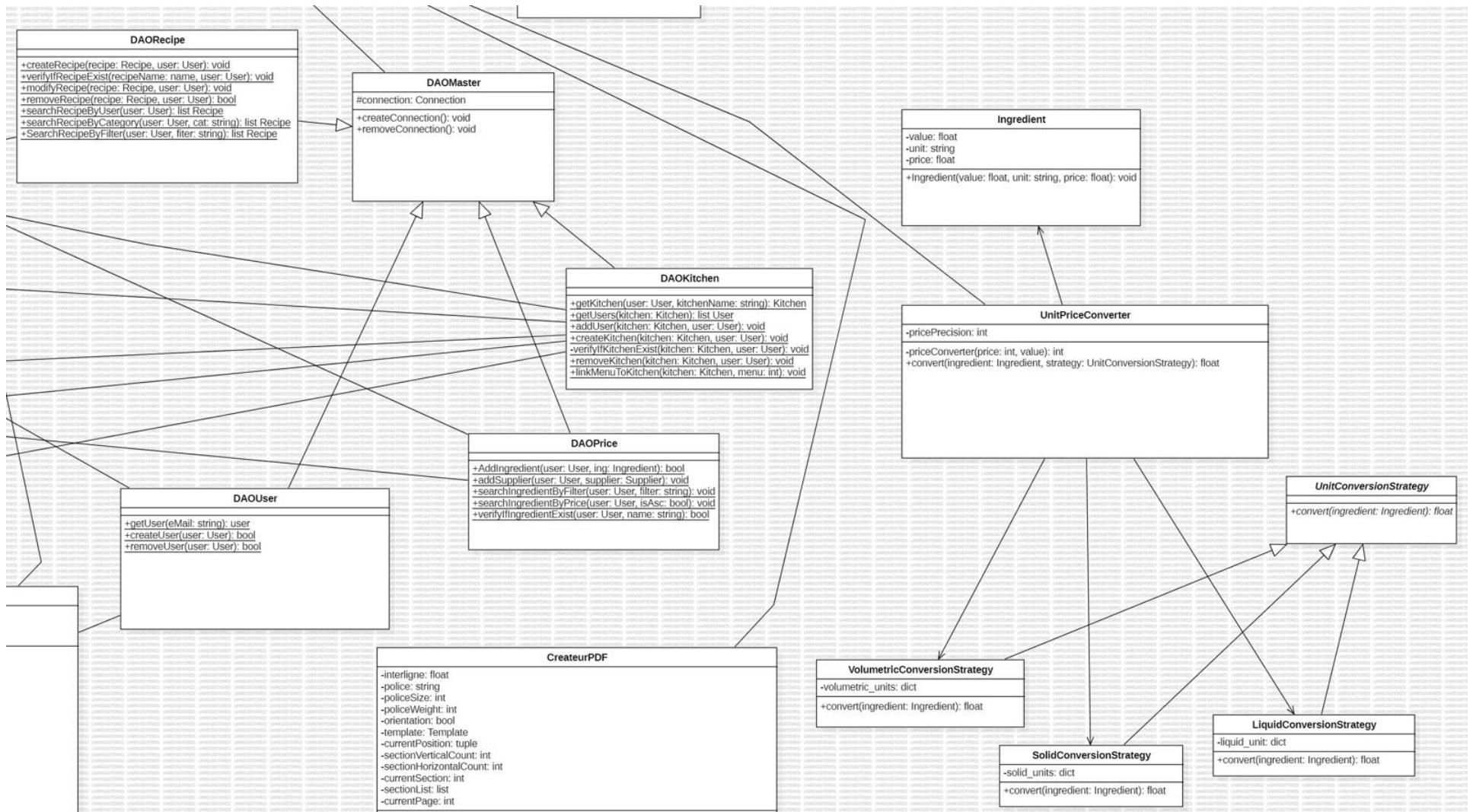




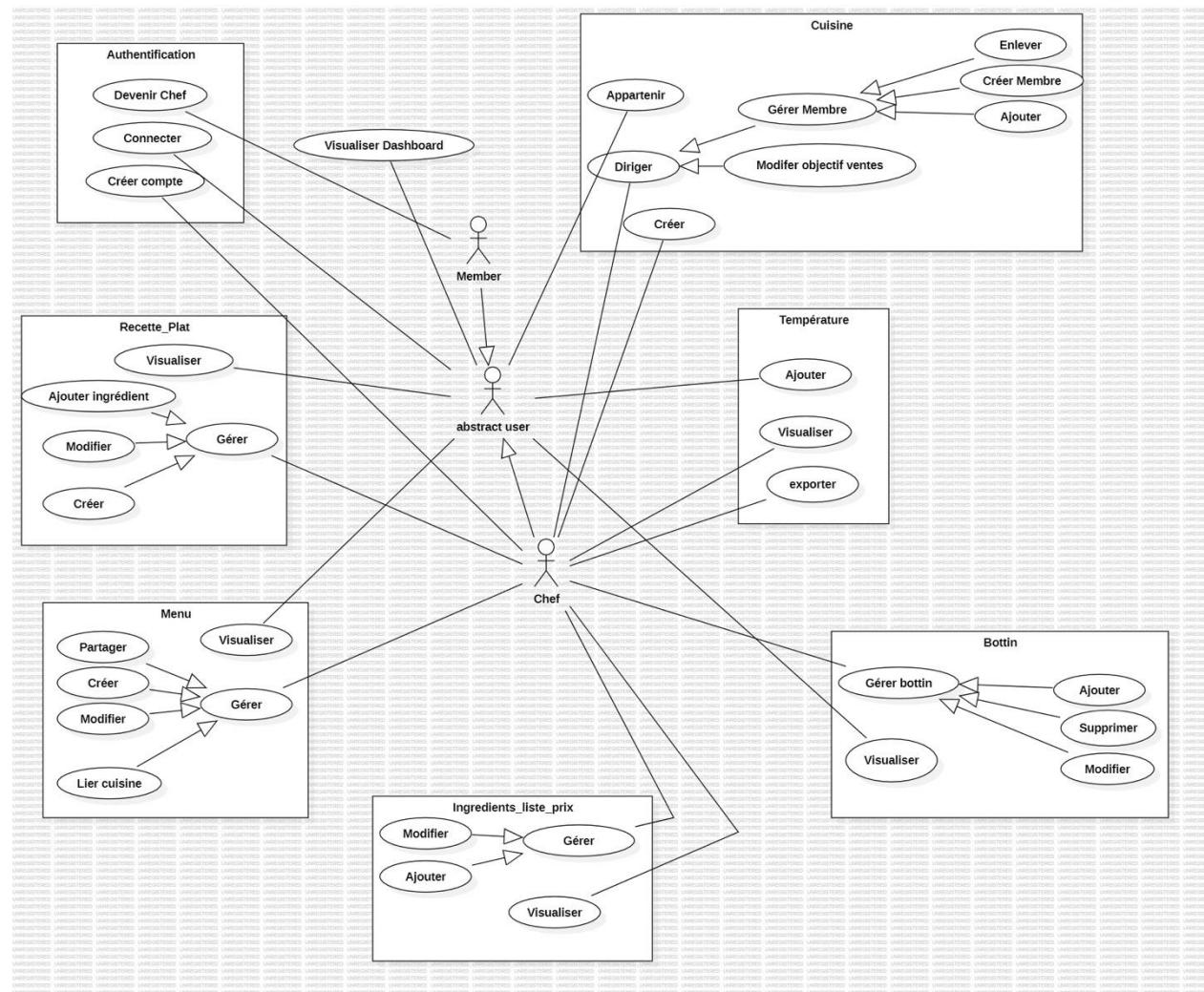








Cas d'utilisation



Annexe

Récurivité :

- Un concept en programmation où une fonction s'appelle elle-même de manière répétée jusqu'à ce qu'une condition de sortie soit atteinte. Elle est souvent utilisée pour traiter des structures de données récursives comme les arbres ou les listes chaînées.

ORM Prisma :

- Un outil de mapping objet-relationnel (ORM) pour les bases de données SQL. Il simplifie l'interaction avec la base de données en générant automatiquement du code JavaScript pour effectuer des opérations CRUD (Create, Read, Update, Delete).

Typage Statique :

- Un système dans lequel les types de variables sont vérifiés lors de la compilation du code plutôt qu'à l'exécution. Cela permet de détecter les erreurs de typage plus tôt dans le processus de développement.

Cluster :

- Un ensemble de serveurs interconnectés qui travaillent ensemble pour fournir des ressources informatiques distribuées, telles que le stockage de données ou la puissance de calcul.

GitHub Actions :

- Outil de déploiement et d'intégration permettant l'automatisation des tests unitaires.

Jest :

- Librairie de test JavaScript.