

# Dossier de Projet



**GoodFood**

Présenté et soutenu par  
**Dorian Guignard**

En vue de l'obtention du  
**Titre Professionnel**  
**Developpeur Web Et Web Mobile**



Centre de Formation **O'Clock** [ Promotion Lucy 2022-2023 ]



Référent de formation : **Ludovic Argenty et Adélie Castel**



# SOMMAIRE

<b>I.</b>	<b><u>INTRODUCTION</u></b>	<b>4</b>
<b>II.</b>	<b><u>LISTE DES COMPETENCES</u></b>	<b>5</b>
A.	DEVELOPPER LA PARTIE FRONT-END D'UNE APPLICATION WEB OU WEB MOBILE EN INTEGRANT LES RECOMMANDATIONS DE SECURITE	5
B.	DEVELOPPER LA PARTIE BACK-END D'UNE APPLICATION WEB OU WEB MOBILE EN INTEGRANT LES RECOMMANDATIONS DE SECURITE	5
<b>III.</b>	<b><u>RÉSUMÉ DU PROJET</u></b>	<b>7</b>
<b>IV.</b>	<b><u>CAHIER DES CHARGES</u></b>	<b>8</b>
1.	LE MVP (MINIMUM VISIBLE PRODUCT)	8
2.	LA V2	9
3.	LE PUBLIC VISÉ	9
4.	DEFINITION DES ROLES ATTRIBUÉES	10
5.	EXEMPLES DE WIREFRAMES	11
6.	ARBORESCENCE DE L'APPLICATION	13
7.	USER STORIES EDITEES SUR TRELLO ET TABLEAU DES FONCTIONNALITES	14
8.	MCD DU PROJET	17
9.	DICTIONNAIRE DE DONNÉES	18
10.	LISTE DES ROUTES	20
11.	CHARTRE GRAPHIQUE	21
<b>V.</b>	<b><u>SPÉCIFICATIONS TECHNIQUES DU PROJET</u></b>	<b>22</b>
A.	L'ARCHITECTURE DU PROJET	22
B.	LE VERSIONNING	23
C.	LES LANGAGES, FRAMEWORKS ET OUTILS UTILISES DANS LE PROJET	26
1.	LES TECHNOLOGIES EMPLOYES SUR L'ENSEMBLE DU PROJET	26
2.	TECHNOLOGIES EMPLOYES COTE BACKEND	27
3.	TECHNOLOGIES EMPLOYES COTE FRONTEND	28
4.	LES NAVIGATEURS COMPATIBLES	28
A.	LES MESURES DE SECURITE APORTEES AU PROJET	29
1.	LA SECURITE COTE BACK END	30
2.	SECURITE COTE FRONT END	35



<b><u>VI. GESTION DE PROJET</u></b>	<b><u>39</u></b>
A. PRESENTATION DE L'EQUIPE ET EXPLICATION DES ROLES DE CHACUN	39
B. METHODOLOGIE DE TRAVAIL	40
C. LES OUTILS UTILISES	42
D. LES SPRINTS	43
<b><u>VII. MES REALISATIONS</u></b>	<b><u>44</u></b>
A. SPRINT 0	44
B. SPRINT 1	47
C. SPRINT 2	51
D. SPRINT FINAL	57
<b><u>VIII. PRESENTATION DU JEU D'ESSAI</u></b>	<b><u>61</u></b>
<b><u>IX. VEILLE SUR LES VULNERABILITE DE SECURITE</u></b>	<b><u>67</u></b>
<b><u>X. PROBLEMES RENCONTRES ET RESOLUTIONS</u></b>	<b><u>69</u></b>
<b><u>XI. EXTRAIT D'UNE DOCUMENTATION D'UN SITE EN ANGLAIS</u></b>	<b><u>70</u></b>
<b><u>XII. CONCLUSION</u></b>	<b><u>72</u></b>
<b><u>ANNEXES</u></b>	<b><u>73</u></b>
1. Rendu visuel GoodFood	74
2. Wireframe réalisé par l'ensemble de l'équipe.	77

## I. INTRODUCTION

Intéressé par le domaine des nouvelles technologies et de l'informatique depuis longtemps, j'ai décidé de me reconvertis dans le développement web en 2022. Après avoir travaillé 10 ans en tant qu'infirmier diplômé d'Etat, je m'apprête à devenir développeur Web Back-end à l'issu d'une formation certifiante avec l'école O'Clock, labellisée « Grande École du Numérique ».

Fort de 700h de formation pour acquérir les compétences de bases du développement Web (technologies HTML5, CSS3, JavaScript, PHP et MySQL), puis une spécialisation Back-end sur Symfony, suivi d'un projet de fin de formation permettant la mise en pratique de l'ensemble des compétences apprises, le métier de développeur Web me passionne.

Le projet qui sera présenté est un exercice de fin de formation permettant de mettre en application les compétences acquises durant la formation, de travailler en groupe selon la méthode AGILE SCRUM, d'utiliser différents outils d'entraide et de développer ses connaissances.

## II. LISTE DES COMPETENCES

### TECHNIQUES COUVERTES PAR LE PROJET

#### A. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

##### ❖ CP1 - Maquetter une application

Lors de la phase de conception nous avons réalisés les Wireframes de l'application avec Whimsical. Pour ce faire nous avons réalisé deux prototypes chacun un en version desktop et l'autre en version smartphone concernant les pages du front-office ainsi que du back-office.

Ensuite nous nous sommes concerté pour élaborer le MVP et réfléchie aux éventuelles fonctionnalité qui pourront être travaillé sur une version 2.

##### ❖ CP2 - Réaliser une interface utilisateur web statique et adaptable

L'interface utilisateur a été conçue sur la base des wireframes et avec une charte graphique définie lors de la phase de conception, en concertation avec l'équipe. Concernant la partie front office, l'intégration web a été réalisée en utilisant les langages HTML et CSS pour élaborer une interface utilisateur statique et cohérente, adaptée à différents formats.

Pour la partie back office nous avons utilisé en partie la librairie Bootstrap pour améliorer l'apparence de l'interface administrateur.

##### ❖ CP3 - Développer une interface utilisateur web dynamique

Le projet GoodFood a été développé avec la librairie REACT en langage Java Script pour une dynamisation de l'application sans recharge de page ainsi qu'avec Ant Design : une bibliothèque d'interface qui contient un ensemble de composants et de fonctionnalité REACT pour créer des interfaces utilisateur riches et interactives.

#### B. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

##### ❖ CP5 - Créer une base de données

La base de données relationnelle a été initialement créé via l'interface graphique Adminer installé sur notre environnement de travail. Les caractéristiques, tables et attributs (issu de la phase de conception : Model Conceptuel de Données et dictionnaire de données) ont été implémenté via l'ORM Doctrine du Framework Symfony.



❖ CP6 - Développer les composants d'accès aux données

La manipulation de données et la récupérations de données se fait directement via l'ORM Doctrine de Symfony à partir d'un système de migration, de POO(programmation Orienté Objet) avec Entity et Repository et avec l'aide de Controller et de Vues adéquates selon une architecture MVC (Model Vue Controller).

❖ CP7 - Développer la partie back-end d'une application web ou web mobile

Pour la réalisation de notre projet, il a été nécessaire de mettre en place un back-office permettant la mise à jour des données des éléments des profils membres ainsi que des différents items tels que les recettes, les ingrédients, les vertus, etc. De plus , certaines fonctionnalités ont été ajoutées pour l'administrateur, notamment la sécurisation des accès et le téléchargement d'image. Nous avons également mis en place une API pour connecter les données entre la base de données située sur le serveur côté back-office et le front-office côté client ; et assuré la sécurisation de cette API en utilisant le protocole JWT Token.

### III. RÉSUMÉ DU PROJET

Le projet GoodFood est né à la fois d'un intérêt pour la nutrition et pour la cuisine. C'est de part c'est deux thématiques que m'est venu l'idée de développer une application pouvant répondre à des attentes en matière de bien-être et d'hygiène de vie.

Initialement, mon objectif était de créer un catalogue de recette 'healthy' permettant aux visiteurs de sélectionner une recette et d'en découvrir les informations nutritionnelles ainsi que les vertus de chaque ingrédients. Ceci afin qu'un visiteur puisse comprendre ce qu'il mange et progresser en matière de ses connaissances dans le domaine de la nutrition et de plus afin qu'il puisse progressivement améliorer son hygiène alimentaire au quotidien.

Cependant, après réflexion d'équipe au cours de la première semaine de conception, il est apparu clairement que le volume du projet était surdimensionné pour le délai accordé à la durée de production. C'est pourquoi nous avons décidé conjointement de délimiter le concept autrement, en imaginant un site permettant aux visiteurs de sélectionner une recette en fonction d'un type de bienfait ou de vertu recherché et non plus en fonction des ingrédients.

Ainsi le projet GoodFood présente un livre de recettes saines et équilibrées qui permet aux utilisateurs de rechercher des recettes en fonction des vertus associées. L'application offre un choix de huit vertus différentes, notamment vitaminée, rajeunissante, détox, boost, protéinée, transit, immunité et sommeil.

L'objectif de cette application est de répondre à la demande croissante de personnes cherchant à améliorer leur hygiène de vie en adoptant une alimentation saine et équilibrée. GoodFood propose des recettes saines et savoureuses qui aident les utilisateurs à améliorer leur équilibre alimentaire et à trouver des options de repas adaptées à leurs besoins en matière de santé.

En tant que Product Owner et développeur back, j'ai supervisé le projet, participé à la conception et développé la partie back end. Le dossier présentera l'intégralité du projet GoodFood avec le cahier des charges, la conception et le développement du projet et enfin le déroulement du projet.

## IV. CAHIER DES CHARGES

### 1. Le MVP (Minimum Viable Product)

Le MVP (Minimum Viable Product) du projet GoodFood comprend plusieurs fonctionnalités clés pour offrir une expérience utilisateur fluide et intuitive. En haut de chaque page du site, un menu de navigation est disponible avec les éléments suivants :

- Un menu permettant à l'utilisateur de naviguer vers les recettes en fonction de leur catégorie et de leur vertu.
- Un bouton de connexion qui redirige vers la page de connexion pour les utilisateurs enregistrés.
- Une barre de recherche pour rechercher une recette spécifique.

En bas de chaque page, des liens vers les mentions légales et un formulaire de contact sont disponibles. Lorsqu'un administrateur se connecte, il aura également accès à un bouton pour accéder aux pages d'administration du site, comme expliqué ci-dessous.

- Sur la page d'accueil, un slider permet de faire défiler une sélection de recettes de droite à gauche. En dessous, des boutons sont disponibles pour filtrer les recettes en fonction de la vertu recherchée.
- En utilisant le menu de navigation, les utilisateurs accèdent aux listes de recettes, apparaissant sous forme de grille de tuiles avec leur image et leur titre. Les recettes peuvent être consultées en utilisant une pagination de droite à gauche.

Lorsqu'un utilisateur clique sur le titre ou l'image d'une recette, il est redirigé vers une page détaillée de la recette, qui affiche l'image de la recette, ses ingrédients et les différentes étapes de sa réalisation.

- Les visiteurs peuvent se connecter au site afin d'ajouter des recettes. Pour cela, en cliquant sur le bouton de connexion dans le menu de navigation, ils sont redirigés vers une page de connexion où ils ont deux options : s'inscrire en utilisant une adresse e-mail, un pseudo et un mot de passe, ou se connecter en utilisant leur adresse e-mail et leur mot de passe.

- Les visiteurs ont également la possibilité de contacter l'administrateur en cliquant sur le lien "contactez-nous", ce qui les redirige vers une page de contact contenant un champ pour taper un message et indiquer une adresse e-mail pour être recontacté.

La partie back-end du projet comprend plusieurs fonctionnalités clés, notamment :

- une page listant les différentes tables/entités de la base de données,
- la gestion de la sécurité avec ACL (Access Control List), CSRF (Cross-Site Request Forgery) et JWT Token (JSON Web Token),
- la conception de la base de données avec MCD (Modèle Conceptuel de Données), MLD (Modèle Logique de Données) et MOCODO (Modèle Conceptuel des Objets et des Données Orientés),
- la mise en place de formulaires avec restrictions et unicité des champs, la création des opérations CRUD (Create, Read, Update, Delete) pour les entités Utilisateur et Recette/Vertus,
- le mapping des entités avec la base de données, et la mise en place d'une API pour la communication entre le front-end et le back-end du site.

## 2. LA V2

Pour la deuxième version du site GoodFood, il est prévu de mettre en place les éléments suivants :

1. Mise en place des Slugs pour inclure le nom de chaque recette dans l'URL afin d'améliorer la lisibilité.
2. Enregistrement des favoris en session utilisateur pour permettre aux utilisateurs de sauvegarder leurs recettes préférées.
3. Ajout d'un espace de commentaire et de notation sur chaque recette pour permettre aux utilisateurs de laisser des commentaires et des évaluations sur les recettes.
4. Indication du nom de la vertu associée à chaque recette sur la page de la recette pour mettre en avant les bienfaits des ingrédients utilisés.
5. Création d'une page de description des vertus pour fournir des informations détaillées sur les bienfaits des ingrédients utilisés dans les recettes.

## 3. LE PUBLIC VISÉ

Toutes personnes soucieuses de son alimentation et souhaitant préparer une recette selon une vertu ou un bienfait recherché.

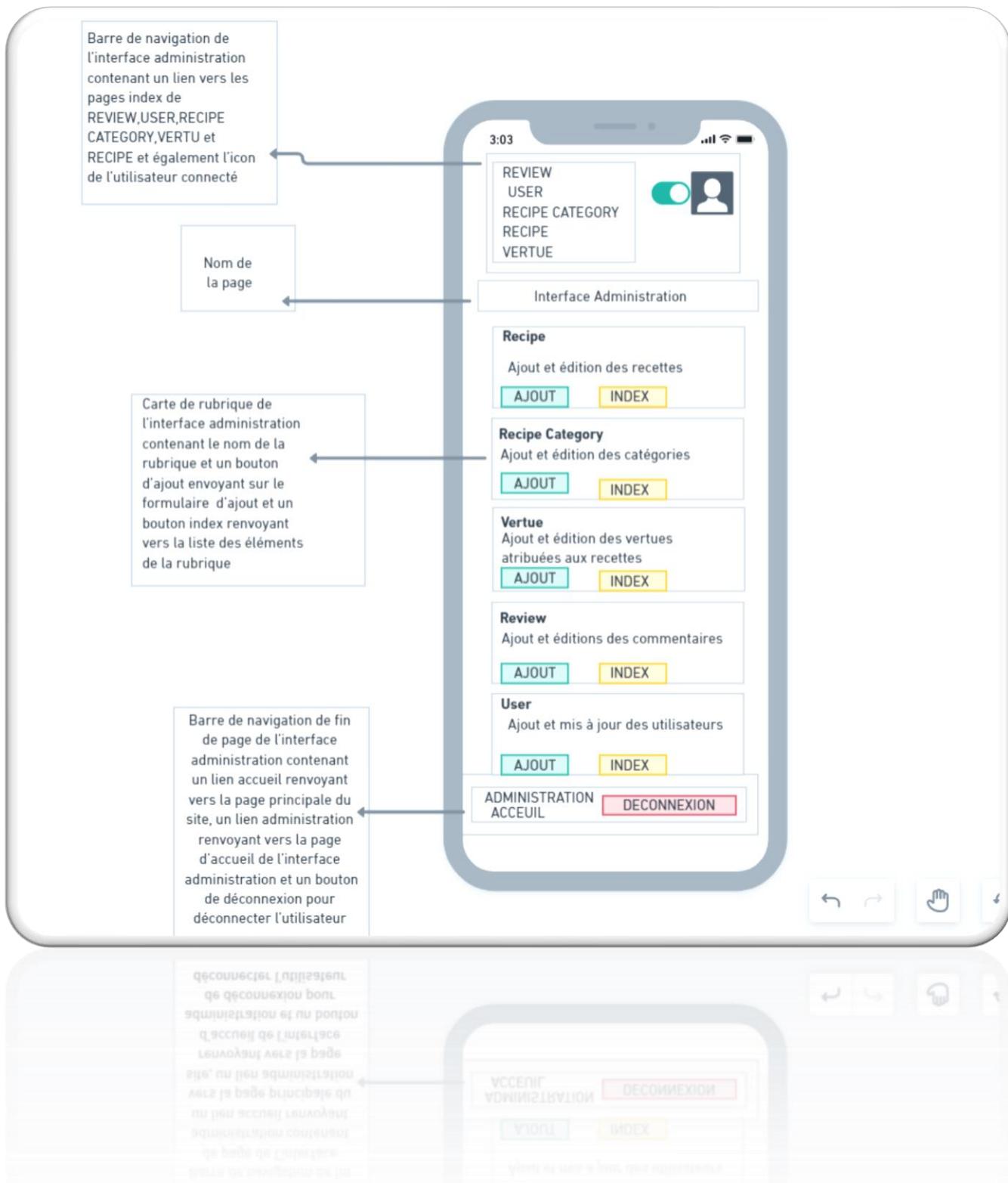
## 4. DEFINITION DES ROLES ATTRIBUÉES

Dans l'application web GoodFood 3 roles sont prédéfinis :

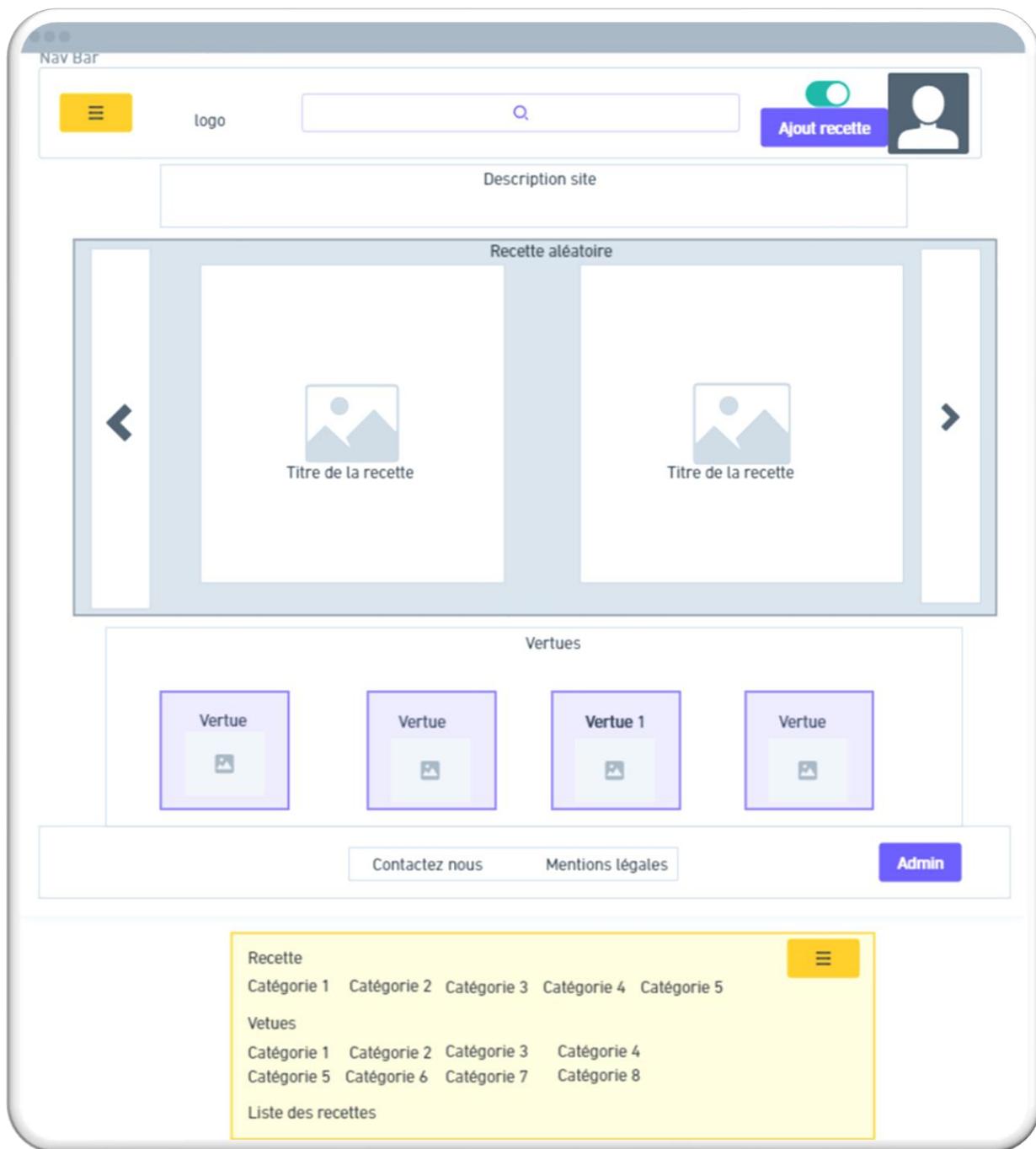
- **Visiteur** : Il s'agit d'un utilisateur qui accède à l'application sans avoir créé de compte. Il peut avoir un accès limité aux fonctionnalités de l'application mais ne peut pas sauvegarder ses préférences.
- **Utilisateur inscrit** : est un utilisateur qui a créé un compte sur l'application. Il a accès à toutes les fonctionnalités disponibles pour les utilisateurs, y compris la sauvegarde de ses préférences et l'historique de ses commandes.
- **Administrateur** : est utilisateur ayant des priviléges spécifiques pour gérer et administrer l'application via le backoffice notamment. L'administrateur peut ajouter, supprimer et modifier des éléments tels que les recettes, les catégories, des comptes utilisateurs, etc.

## 5. EXEMPLES DE WIREFRAMES

Wireframe version mobile pour le back-office



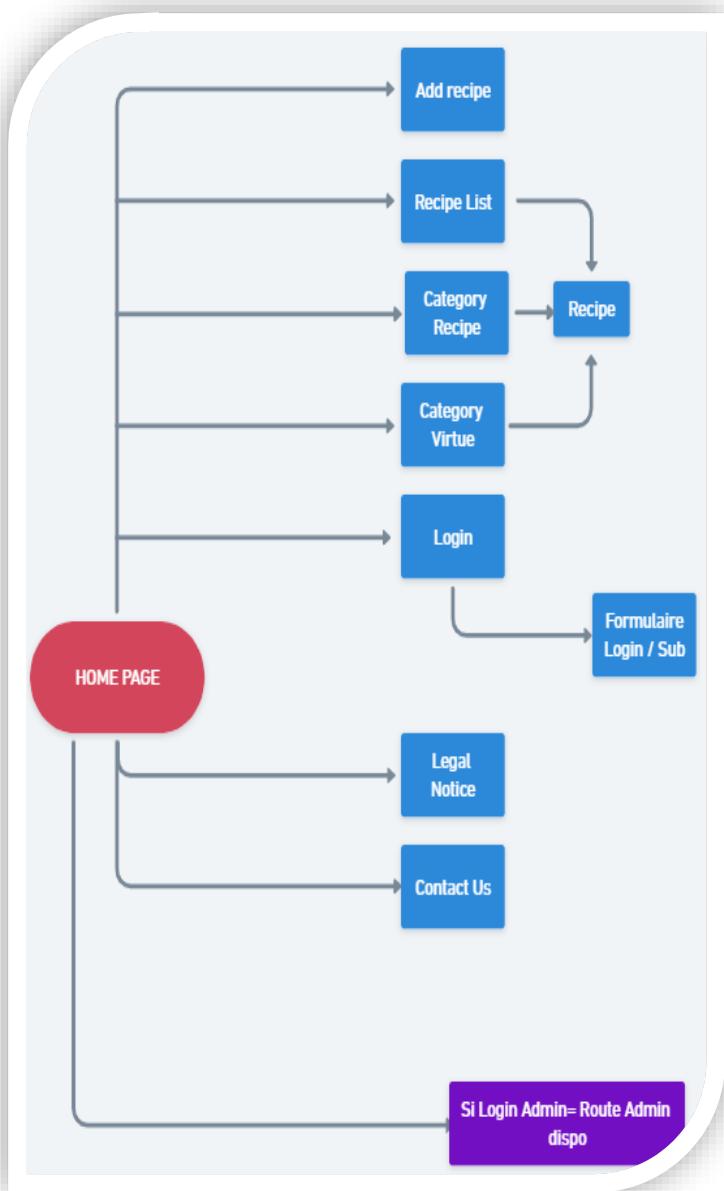
## Wireframe version desktop pour le front-office page d'accueil



→ La liste complète des Wireframes se trouve en annexes du dossier.



## 6. ARBORESCENCE DE L'APPLICATION



## 7. User Stories éditées sur Trello et Tableau des fonctionnalités

The screenshot shows a Trello board titled "My Recette". The board has three columns:

- À faire** (To Do):
  - En tant qu'Admin, je veux ajouter une recette
  - En tant qu'Admin, je veux supprimer une recette
  - En tant qu'Admin, je veux pouvoir modifier une recette
  - En tant qu'Admin , je veux ajouter un ingrédient
  - En tant qu'Admin, je veux supprimer un ingrédient
  - En tant qu'Admin, je veux pouvoir supprimer un Utilisateur
  - En tant qu'Admin , je veux ajouter un utilisateur
  - En tant qu'Admin je veux ajouter une vertue
  - En tant qu'Admin je veux pouvoir modifier une vertue
  - En tant qu'admin je veux pouvoir supprimer une vertue
  - En tant qu'Admin, je veux me connecter
- À faire**:
  - En tant que visiteur/utilisateur je veux voir la liste des recettes par vertue
  - En tant que visiteur/utilisateur, je veux voir la liste des recettes par catégories
  - En tant que visiteur/utilisateur je veux pouvoir passer le site en mode sombre
  - En tant que visiteur/utilisateur je veux pouvoir consulter les mentions légales du site
  - En tant que visiteur/utilisateur, je veux faire une recherche pour trouver une recette
  - En tant que visiteur/utilisateur, je veux voir la sélection des recettes aléatoires
- À faire**:
  - En tant qu'Utilisateur, je veux ajouter une recette
  - En tant qu'utilisateur, je veux modifier ma recette créée
  - En tant que Visiteur/Utilisateur, je veux voir la liste des recettes
  - En tant que Visiteur, je veux pouvoir me créer un compte
  - En tant que Visiteur/Utilisateur , je veux pouvoir contacter l'Admin
  - En tant qu'Utilisateur , je veux me connecter
  - En tant qu'utilisateur je veux pouvoir choisir un pseudo lors de mon inscription
  - En tant que visiteur/utilisateur je veux consulter une recette ses ingrédients et sa préparation

Each column has a "Ajouter une carte" (Add a card) button at the bottom.



⇒ EN TANT QU'ADMIN

En tant que	Je veux	Afin de
<b>Admin</b>	Lire, ajouter, modifier ou supprimer une recette	Pouvoir interagir sur une recette créée
<b>Admin</b>	Lire, ajouter, modifier ou supprimer un ingrédient	Pouvoir interagir sur un ingrédient créée
<b>Admin</b>	Lire, ajouter, modifier ou supprimer une catégorie	Pouvoir interagir sur une catégorie créée
<b>Admin</b>	Lire, ajouter, modifier ou supprimer une vertu	Pouvoir Interagir sur une vertu créée
<b>Admin</b>	Ajouter ou supprimer un utilisateur	Gerer les utilisateurs
<b>Admin</b>	Me connecter au backoffice via l'interface du FrontOffice	Modifier l'interface et les données du front office
<b>Admin</b>	Me déconnecter du backoffice et revenir sur le front office	Quitter l'application
<b>Admin</b>	Telecharger une image via les formulaires du backoffice	Intégrer une image sur un élément souhaité



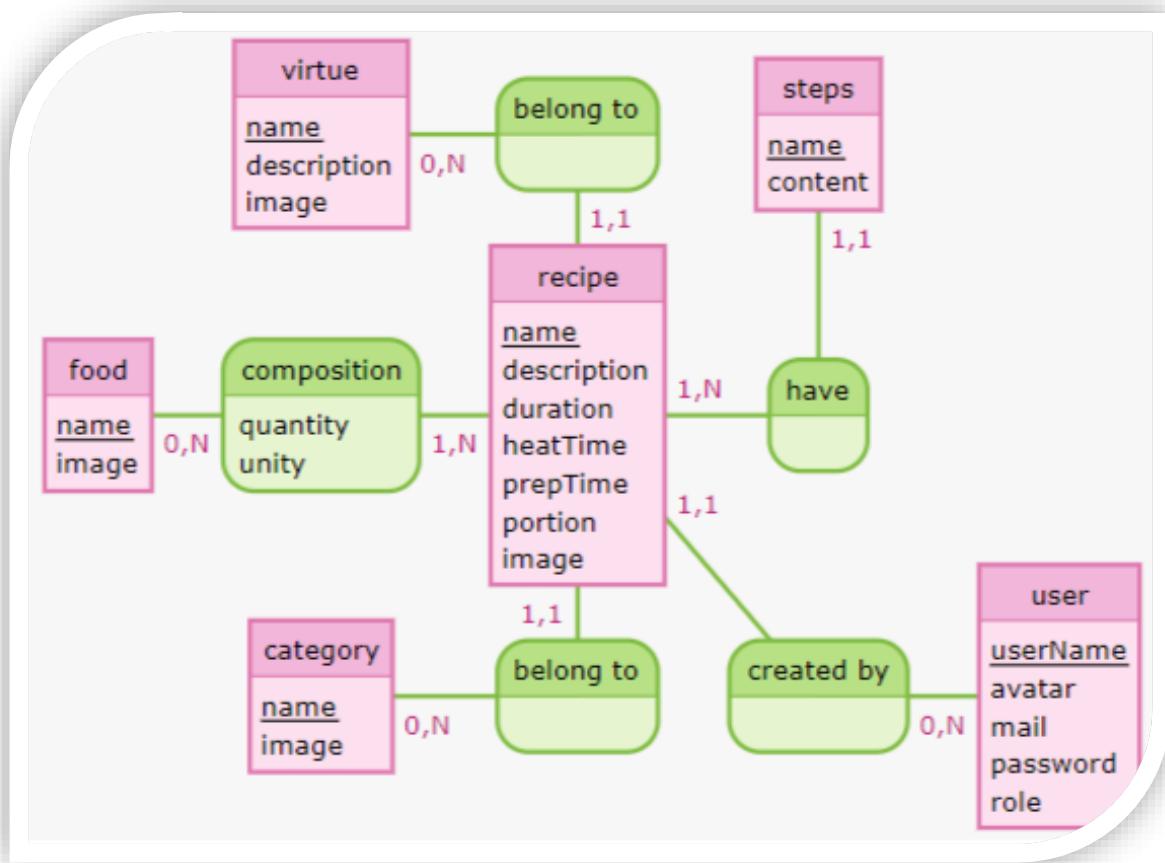
⇒ EN TANT QUE VISITEUR OU UTILISATEUR INSCRIT

En tant que	Je veux	Afin de
Visiteur/Utilisateur	Voir la liste des recettes par vertu	Selectionner une recette en fonction du type de bienfait
Visiteur/Utilisateur	Voir la liste des recettes par catégorie	Selectionner des recettes en fonction du type de repas
Visiteur/Utilisateur	Consulter les mentions légales du site	
Visiteur/Utilisateur	Contacter l'administrateur du site	Poser une question en fonction d'un besoin précis
Visiteur/Utilisateur	Voir la sélection des recettes aléatoires	D'avoir un visu du type de recettes présentes sur l'application
Visiteur/Utilisateur	Faire une recherche pour trouver une recette par nom ou par ingrédient	Trouver une recette plus facilement
Visiteur/Utilisateur	Pouvoir me créer un compte	D'être inscrit sur le site
Visiteur/Utilisateur	Consulter une recette	La choisir éventuellement
Visiteur/Utilisateur	Revenir sur la page d'accueil	Me rediriger plus facilement
Utilisateur Inscrit	Ajouter, supprimer, modifier ma recette	D'interagir avec une recette sur le site
Utilisateur Inscrit	Me connecter ou me déconnecter	Afin d'accéder à d'autres fonctionnalités ou de quitter l'application
Utilisateur Inscrit	Choisir une image lors de mon inscription	Voir apparaître cette image sur la page principale
Utilisateur Inscrit	Consulter mon profil	Voir mes recettes créées



## 8. MCD DU PROJET

Le model conceptuel de données, présenté ici, est prévu pour représenter les données ainsi que les relations entre les différentes tables, entre un client et une équipe de développement web. C'est le model logique de donnée, puis le model physique de données qui vont permettre aux développeur de produire la base de données. En y indiquant notamment les clés primaires et les clés étrangères ainsi que d'autres informations susceptible d'aider au développement du projet.



## 9. DICTIONNAIRE DE DONNÉES

Table Recipe			
Champ	Type	Spécificités	Description
code_recipe	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO	L'ID de la recette
name	VARCHAR(64)	NOT NULL	Le nom de la recette
description	TEXT	NOT NULL	La description de la recette
time	TIME	NOT NULL	Le temps de préparation de la recette
number	INT	NOT NULL	La quantité en nombre de personnes
picture	VARCHAR (255)	NOT NULL	L'image d'illustration
code_user	ENTITY	PRIMARY KEY,UNSIGNED,NOT NULL	L'ID de l'utilisateur
code_category	ENTITY	PRIMARY KEY,UNSIGNED,NOT NULL	ID catégorie de la recette
code_vertue	ENTITY	PRIMARY KEY,UNSIGNED,NOT NULL	ID description de la vertue

Table Vertue			
Champ	Type	Spécificités	Description
code_vertue	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO	L'ID de la vertue
name	VARCHAR (64)	NOT NULL	le nom de la vertue
description	VARCHAR (255)	NULLABLE	la description de la vertue
picture	VARCHAR (255)	NULLABLE	L'image de la vertue

Table User			
Champ	Type	Spécificités	Description
code_user	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO	L'ID de l'utilisateur
nameuser	VARCHAR (64)	NOT NULL	le nom de l'utilisateur
picture	VARCHAR (255)	NULLABLE	l'image de l'utilisateur
password	VARCHAR (255)	NOT NULL	mot de passe de l'utilisateur
email	VARCHAR (255)	NOT NULL	l'adresse mail de l'utilisateur



Table Intermediaire Composition			
Champ	Type	Spécificités	Description
code_food	ENTITY	PRIMARY KEY,UNSIGNED,NOT NULL	L'ID de l'aliment
code_recipe	ENTITY	PRIMARY KEY,UNSIGNED,NOT NULL	L'ID de la recette
quantity	INT	NOT NULL, UNSIGNED	Quantité de l'ingrédient
unity	VARCHAR (20)	NULLABLE	Unité de l'ingrédient
Table Food			
Champ	Type	Spécificités	Description
code_food	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO	L'ID de l'aliment
picture	VARCHAR (255)	NULLABLE	l'image de l'aliment
name	VARCHAR(64)	NOT NULL	Le nom de l'aliment
Table Category			
Champ	Type	Spécificités	Description
code_category	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO	L'ID du category_recipe
name	VARCHAR(64)	NOT NULL	Le nom de la category_recipe
picture	VARCHAR (255)	NULLABLE	l'image de la category_recipe
Table Steps			
Champ	Type	Spécificités	Description
code_steps	INT	PRIMARY KEY, UNSIGNED, NOT NULL, AUTO	L'ID de steps
name	INT	NOT NULL	Le numéro de l'étape
content	VARCHAR(255)	NOT NULL	Contenu de l'étape



## 10. LISTE DES ROUTES

Partie	Page	Route
Front Office	Page d'accueil	/home
Front Office	Page recette	/recipe/{id}
Front Office	Page ajout recette	/recipe/add
Front Office	Page recettes en fonction des vertus	/virtue/{id}
Front Office	Page recettes en fonction des catégories	/category/{id}
Front Office	Page Login	/login
Front Office	Page profil	/pseudo
Front Office	Page mentions	/notice
Front Office	Page Contact	/contact
Back Office	Page d'accueil	/backhome
Back Office	Page listes	/index/entity
Back Office	Page d'édition	/edit/entity
Back Office	Page new	/new/entity
Back Office	Page détail	/show/entity
API	Afficher la liste avec les détails	GET /api/nameEntity
API	Afficher les détails d'une entité	GET /api/nameEntity/{id}
API	Création d'une entité	POST /api/nameEntity/add
API	Modification d'une entité	PATCH /api/nameEntity/{id}
API	Suppression d'une entité	DELETE /api/nameEntity/{id}

NB : "nameEntity" ou "entity" doivent être remplacés par le nom de l'entité spécifique.

## 11.CHARTRE GRAPHIQUE



- Couleur choisie pour le texte et les boutons de la nav-bar (Titre du Site, Loupe, Connexion) et du footer



- Couleur choisie pour les titres du menu burger (recettes par catégories...) , ainsi que le hover des autres textes (nom des catégories)



- Couleur choisie pour les textes principaux ( description d'une recette, titre des recettes )



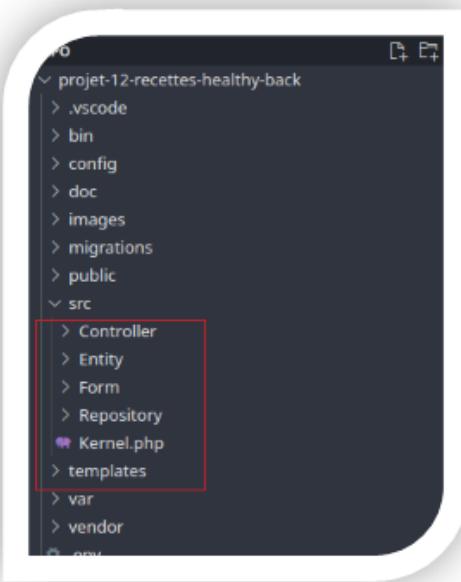
## V. SPÉCIFICATIONS TECHNIQUES DU PROJET

### A. L'architecture du projet

Le projet GoodFood se compose de deux répositorie, ainsi l'un comprend la partie backend développer à partir du framework Symfony et l'autre la partie frontend développer principalement avec la bibliothèque REACT.

La partie backend possède une architecture MVC (Modèle-Vue-Contrôleur) pour organiser et structurer le code. Ainsi pour convenir au modèle MVC nous avons mis en place et travailler depuis la structure comprenant les dossiers suivant:

- Dossier **Controller**: qui contient les contrôleurs de l'application responsables de la logique de traitement des requêtes HTTP et de la génération des réponses.
- Dossier **Entity**: qui contient les entités de doctrine, classes PHP qui représentent les objets persistants dans la base de données.
- Dossier **Repository**: contenant les classes qui sont responsables de l'accès à la base de données pour récupérer et persister les objets.
- Dossier **Form**: qui contient les classes de formulaires de Symfony, qui sont utilisées pour créer et valider les formulaires HTML.
- Et enfin le Dossier **View**: qui contient les templates Twig (moteur de template flexible) utilisés pour générer les vues HTML de l'application.



**La partie front end**, quant à elle, suit une architecture basée sur **la logique de composants** en REACT représentant les différentes fonctionnalités et caractéristiques présentées sur le FrontOffice.

Les deux dépôts communiquent entre eux via **Axios** pour les requêtes émises et reçues côté front-end et via une **API REST** développée coté back-end. De plus un ensemble de configuration de sécurité paramétrées coté backend permettent des échanges sécurisés et fiables que l'on explicitera plus bas.

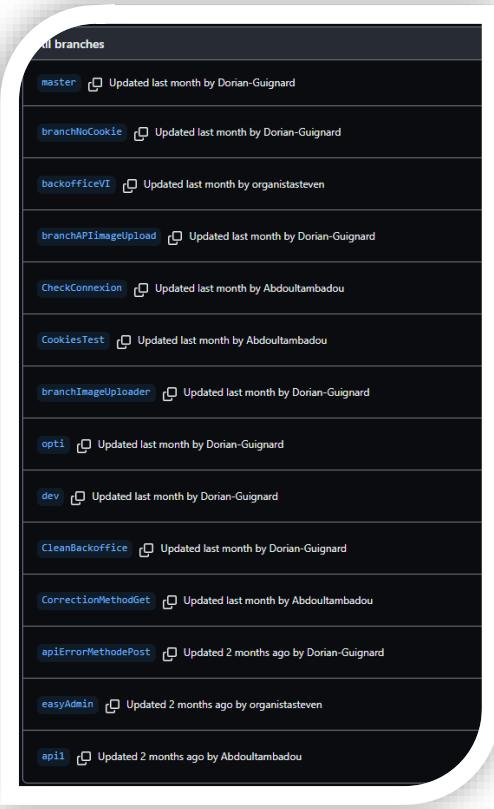
## B. Le Versionning

Concernant le versioning nous avons travaillé sur deux repositories distincts, l'un était utilisé pour la partie front-end et l'autre pour le développement de la partie back-end. La séparation des deux repository nous a permis de travaillé de manière indépendante sur nos fonctionnalités respectives sans avoir a entré en conflit avec l'une ou l'autre des parties.

Concernant le travail sur le repository back-end nous avons créé plusieurs branches afin de développer les fonctionnalités du projet. Nous avons conservé la branche principale 'master' et travailler sur d'autres branches tout au long du projet. En terme d'organisation coté backend nous avons travaillé sur l'intégralité du projet en équipe. En général deux personnes travaillaient sur une branche et la troisième personne travaillait sur une autre branche ou recherchait des informations concernant une problématique ou une fonctionnalité.

Le nom des branches a été données en fonction de la fonctionnalité à développer, d'une action souhaité ou bien concernant une problématique particulière.

## Ensemble des branches créées pour le projet GoodFood :

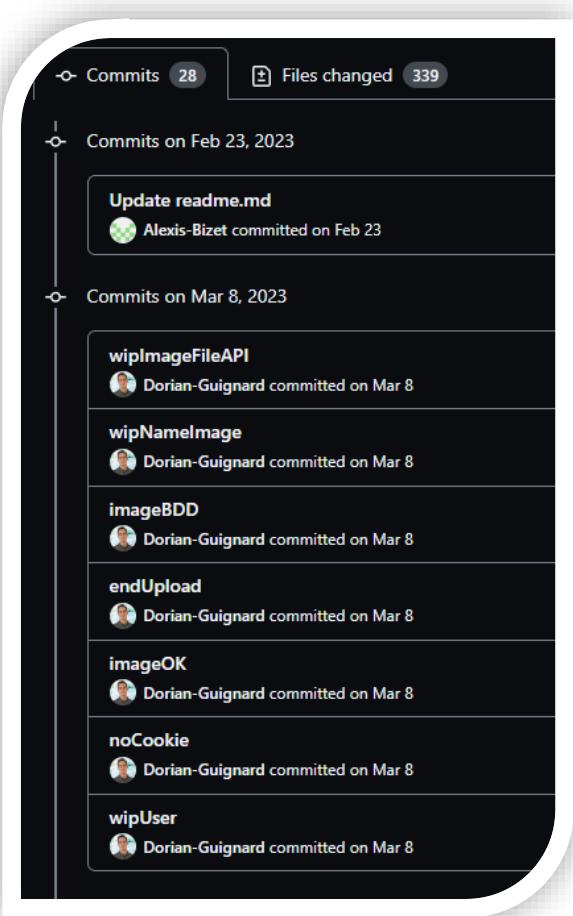


Il est intéressant de noter que nous n'avions pas pensé au départ à la nomenclature à utiliser pour les branches de notre projet. Ce n'est qu'au fur et à mesure de notre progression que nous avons réalisé l'importance de donner des noms cohérents aux différentes branches afin de faciliter le suivi de l'avancement du projet et permettre nous retrouver plus facilement.

De plus nous avons réalisés de nombreux commit. Selon le site <https://www.atlassian.com/fr/git/tutorials/> « les commits constituent les piliers d'une chronologie de projet Git[ . ]. Ils sont créés grâce à la commande git commit pour capturer l'état d'un projet à un point dans le temps. Les instantanés Git sont toujours commités dans le dépôt local. »

Les commits nous ont été indispensables dans l'évolution de notre projet et notamment pour comprendre certaines erreurs occasionnées lors du développement de certaines fonctionnalités. En effet pouvoir comparer certains fichiers grâce à un commit et à un autre, là où il y a eu modification, ajout ou suppression de code nous ont permis de retrouver facilement la source d'un problème généré précédemment.

Exemple de commit que j'ai réalisé concernant la branche : BranchImageUploader



De plus afin de pouvoir répondre aux différents besoins de l'équipe front-end (API, nouvelles données, création d'un utilisateur, ...) nous avons dû mettre régulièrement à jour le répertoire back sur leur environnement de travail. Cette mise à jour peut être réalisé en effectuant dans un premier temps la commande : 'git push' : permettant d'envoyer le code chargé précédemment vers le dépôt centralisé puis avec la commande 'git pull' permettant d'extraire les changements sur son répertoire local. Néanmoins cette action a souvent occasionné des problématiques sur la partie front-end notamment pour la connexion et la cohérence des entités avec la base de données.

Des problèmes liés au système de migrations sont survenus assez fréquemment, notamment après un git pull, où il était souvent difficile de récupérer les données côté front-end car au préalable plusieurs modifications avaient été apportées aux entités Symfony. Par exemple, le changement de type de la propriété 'steps' passant de 'string' à 'integer' dans l'entité 'recipe' a généré un conflit de mapping. En conséquence, l'ORM Doctrine n'était plus en mesure de faire correspondre le code de l'entité 'recipe' avec le type de champ enregistré dans la base de données du front-end, qui n'avait pas été mis à jour au préalable.

## C. Les langages, frameworks et outils utilisés dans le projet

### 1. Les technos employés sur l'ensemble du projet



1. **HTML** (HyperText Markup Language) : est le langage de balisage standard utilisé pour structurer le contenu d'une page web en utilisant des balises pour décrire les éléments tels que les titres, les paragraphes, les images, les liens, etc. Le language HTML a été utilisé tout au long du projet afin de construire la structure des pages web en front end et pour le backoffice en backend.
2. **CSS** (Cascading Style Sheets) : est un langage de style utilisé pour définir la présentation et l'apparence des pages web. Il permet de décrire la mise en page, les couleurs, les polices, les marges, les bordures et autres styles pour embellir et formater les pages web créées avec HTML. Le langage CSS a été utilisé tout comme Twig sur la partie backoffice pour améliorer l'apparence des différentes interfaces et le responsive design. (Le responsive design a été ajouté uniquement sur la partie frontOffice).
3. **Insomnia REST** : est un logiciel qui permet de tester et de déboguer les API REST (Representational State Transfer). Ce logiciel nous a permis entre autre de créer, de gérer et d'envoyer des requêtes HTTP, ainsi que de visualiser les réponses de notre API en temps réel et de déboguer si besoin.
4. **GIT** : est un système de gestion de version décentralisée, utilisé pour le suivi des modifications du code source d'un projet informatique. Il nous a permis de travailler simultanément sur les mêmes fichiers, de fusionner les modifications et de revenir à des versions antérieures en cas de besoin, facilitant ainsi la collaboration et la gestion des versions du code.
5. **GitHub** est une plateforme d'hébergement de code source basée sur Git. Elle offre des fonctionnalités supplémentaires comme la gestion des problèmes, le suivi des modifications, la collaboration entre développeurs, et la possibilité de partager des projets avec la communauté open source. GitHub nous a ainsi permis de collaborer facilement entre tous et à contribuer à la résolution de problème de Versionning notamment.
6. Librairie **Bootstrap** : est une librairie de développement web open source créé par Twitter offrant une collection de composants et de styles prédéfinis permettant de créer facilement des sites web modernes et responsives. Nous l'avons utilisé dans ce projet afin de mettre en place rapidement un back office léger et visuellement agréable ainsi que pour quelques éléments côté front end.



## 2. Technologies employées coté Backend

7. **SQL** (Structured Query Language) : est un langage de programmation utilisé pour gérer les bases de données relationnelles. Il permet de créer, lire, mettre à jour et supprimer des données dans les bases de données, ainsi que de définir et manipuler leur structure.  

8. **SYMFONY** : est un framework PHP open-source utilisé pour développer des applications web modernes et performantes. L'intégralité du projet coté backend a été développé avec le framework et l'architecture offerte par Symfony. Nous avons utilisé la version 5.4 dans ce projet pour correspondre aux enseignements donné lors de la formation.  

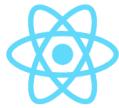
9. **PHP** (Hypertext Preprocessor) : est un langage de script côté serveur principalement utilisé pour développer des applications web dynamiques. Notre projet a été réalisé avec PHP7.4.3 pour une question de compatibilité avec Symfony version 5.4.  

10. **Doctrine** : est un ORM (Object-Relational Mapping) qui permet de simplifier l'interaction entre une application Symfony et une base de données relationnelle. Doctrine permet de manipuler les données de la base de données en utilisant des objets PHP au lieu d'écrire des requêtes SQL en utilisant le paradigme de la programmation associée objet (POO). Nous avons utilisé Doctrine pour le mapping, la manipulation des données, la création d'entité, les migrations ...  

11. **Twig** : est un moteur de template open source pour PHP principalement utilisé dans le contexte du framework Symfony offrant une syntaxe simple pour la création de templates HTML, avec des fonctionnalités telles que l'héritage, l'inclusion de sous-templates, les boucles, les conditions, les filtres... Nous avons utilisé Twig afin de créer des vues dynamiques à notre projet.  

12. **Composer** : est un outil de gestion des dépendances pour PHP. Il permet de gérer facilement les bibliothèques et les dépendances nécessaires dans un projet PHP, en les installant, les mettant à jour et les gérant automatiquement.  


### 3. Technologies employées côté Frontend



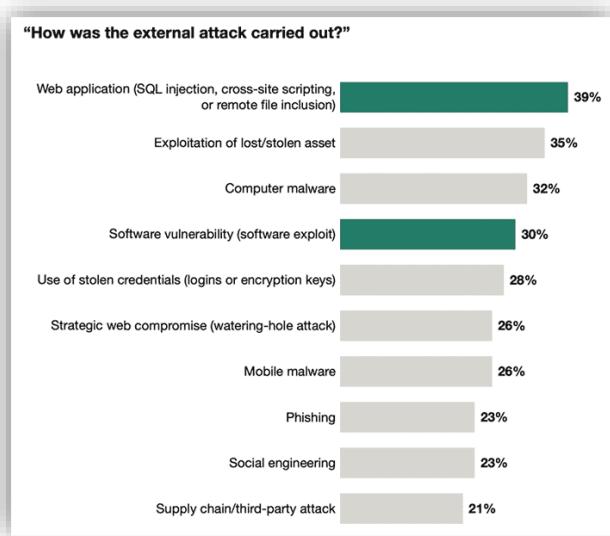
13. **REACT**: est une librairie JavaScript open-source permettant de construire des interfaces utilisateur interactives. L'ensemble du front office a été crée avec REACT afin de manipuler des composants réutilisables et d'organiser efficacement l'interface utilisateur de l'application.
14. **Redux** : est une bibliothèque open-source permettant la gestion d'état dans les applications web qui utilise React pour la construction d'interfaces utilisateur. Redux permet de faciliter la gestion des données et des interactions complexes entre les composants de l'application.
15. **JavaScript (JS)** : est un langage de programmation principalement utilisé côté client pour développer des fonctionnalités interactives sur les sites web. Il permet d'ajouter de l'interactivité, de la dynamique et de la logique aux pages web.
16. **antDesign** : est une librairie de composants d'interface utilisateur open-source pour React. Elle a permis à l'équipe front de créer rapidement des interfaces utilisateur via le site <https://ant.design>
17. **Axios** : est une bibliothèque JavaScript qui a été utilisé dans notre projet afin d'effectuer des requêtes HTTP depuis un navigateur ou depuis un serveur utilisant Node.js, notamment pour effectuer des requêtes GET, POST, PUT, DELETE, etc., ainsi que pour gérer les interceptions de requêtes et les erreurs.
18. **Yarn** : est un gestionnaire de paquets pour les projets JavaScript. Il permet de télécharger et d'installer des dépendances pour un projet, et de les gérer efficacement. Cela permet aussi de garantir que votre application ou votre site web fonctionne correctement et sans erreurs..

### 4. Les navigateurs compatibles



## A. Les mesures de sécurité apportées au projet

D'après le site [yeswedev.bzh](#) « les attaques contre les applications web sont très courantes et de plus en plus fréquentes. Selon le site [Forrester](#) on constate que 39 % de toutes les attaques ont été conçues pour exploiter les vulnérabilités des applications web. »



Le site [yeswedev.bzh](#) nous informe aussi que « les applications web sont particulièrement vulnérables aux attaques, car elles sont exposées à Internet. De nombreux vecteurs d'attaque contre les applications web se concentrent sur la manipulation des entrées utilisateur via des formulaires web et des entrées machine via des API. »

De plus la page [OWASP Top 10](#), du site OWASP référence en matière de sécurité des applications mobiles, répertorie les risques de sécurité les plus critiques pour les applications web. Tels qu'en première position:

1. Les Injections SQL
2. Le Cross-Site Scripting (XSS)
3. Les Inclusion de fichiers à distance (RFI)
4. Le Cross-Site Request Forgery (CSRF)

Ainsi, pour répondre à la demande croissante en matière de cybersécurité, nous avons pris plusieurs mesures de sécurité que nous avons intégrées à notre projet.

## 1. La Sécurité coté Back end

Du coté backend mon équipe et moi avons mis en place plusieurs méthodes permettant de garantir la sécurité des données, de l'utilisateur ou encore de l'application.

- Avec l'installation du composant NelmioCorsBundle nous avons pu implémenter la logique de **CORS** (Cross-Origin Resource Sharing). Cette logique de sécurité permet de contrôler les requêtes HTTP/HTTPS entre différents domaines, par exemple entre le serveur front et le serveur backend du projet.

En l'occurrence la configuration ci-dessous présente dans le fichier **.env** fait en sorte que les requêtes cross-origin soient toutes autorisées (**CORS\_ALLOW\_ORIGIN**) à partir de la chaîne de caractère localhost ou 127.0.0.1 et avec n'importe quel numéro de port. Cela permet ainsi d'autoriser les requêtes provenant d'autres sources.

```
##> nelmio/cors-bundle ##>
CORS_ALLOW_ORIGIN='^https://(localhost|127\.0\.0\.1)(:[0-9]+)?$'
##< nelmio/cors-bundle ##<
```

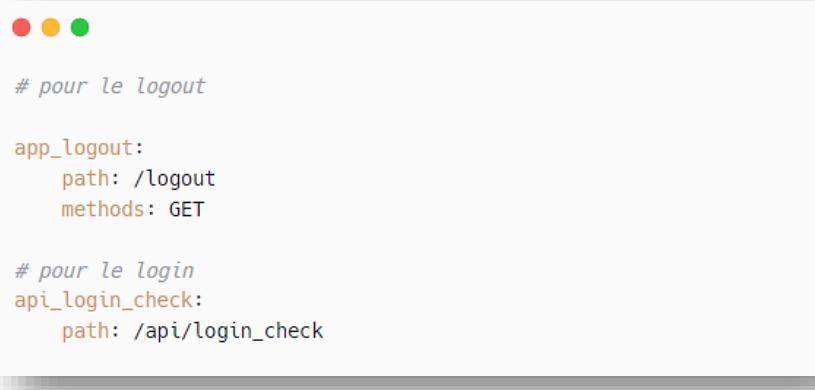
C'est dans le fichier **nelmio\_cors.yaml** que nous avons configuré les paramètres pour les routes de l'application et notamment concernant les requêtes cross origine provenant de la partie frontend via le **localhost:3000**. Cf code ci-dessous :

```
nelmio_cors:
    defaults:
        origin_regex: true
        allow_origin: [%env(CORS_ALLOW_ORIGIN)%,
        'http://localhost:3000']
        allow_methods: ['GET', 'OPTIONS', 'POST', 'PUT', 'PATCH',
        'DELETE']
        allow_headers: ['Content-Type', 'Authorization']
        expose_headers: ['Link']
        max_age: 3600
    paths:
        '^/': null
```

En outre ces paramètres aident à prévenir les attaques entre sites (CSRF) et à sécuriser les interactions entre différents domaines.

- Afin de protéger l'utilisateur nous avons mis en place le système de **JWT-Token** (JSON Web Token). JWT est un format de jeton ou token sécurisé sous forme de chaîne de caractères, utilisé pour l'authentification et l'autorisation dans les applications web et les services API. Nous avons donc utilisé la logique de JWT-Token afin de transmettre et de sécuriser les informations d'identification d'un utilisateur entre le backend et le frontend.
- Pour ça il a fallu installer le composant *lexik/jwt-authentication-bundle*, puis paramétrier certains fichiers de configuration.

Par exemple le code ci-dessous provenant du fichier **routes.yaml** permet grâce à la route **app\_logout** de gérer la déconnexion d'un utilisateur authentifié. De plus il vérifie les informations de connexion (nom d'utilisateur et mot de passe) soumises par un utilisateur grâce à la route **api\_login\_check**.



```
# pour le logout

app_logout:
    path: /logout
    methods: GET

# pour le login
api_login_check:
    path: /api/login_check
```

Le code ci-dessous issu du fichier **lexik\_jwt\_authentication.yaml** configure l'authentification JWT. Il permet de préciser les chemins d'accès aux clés privées et publiques utilisées pour signer et vérifier les jetons JWT, la phrase secrète pour déverrouiller la clé privée, le champ d'identité de l'utilisateur et la durée de validité des jetons.



```
lexik_jwt_authentication:
    secret_key: '%kernel.project_dir%/config/jwt/private.pem'
    public_key: '%kernel.project_dir%/config/jwt/public.pem'
    pass_phrase: '%env(JWT_PASSPHRASE)%'
    user_identity_field: email
    token_ttl: 21600
```

Enfin la génération de clé publique et privée dans le projet permet de garantir que les tokens ne soient pas modifiés ou falsifiés lorsqu'ils sont utilisés pour l'authentification ou l'autorisation dans une application.

- **L'ACL** (Access Control List) est un mécanisme utilisé pour gérer les autorisations d'accès aux ressources et déterminer qui est autorisé à effectuer quelles actions sur ces ressources. Pour configurer une ACL, il s'agit de définir les entrées de contrôle d'accès, qui précisent le sujet (utilisateur ou groupe d'utilisateurs), la ressource et le type d'accès autorisé.

Le code ci-dessous présent dans le fichier security.yaml permet en fonction du rôle de l'utilisateur d'accéder ou non aux divers routes du backoffice. Il définit aussi les accès à l'API développer dans le projet. Ainsi 'path' correspond à l'url, 'roles' étant les rôles attribuées ayant accès ou pas et 'methods' est la méthode qui sera acceptée pour la requête.



```
access_control:
# - { path: ^/admin, roles: ROLE_ADMIN }
# - { path: ^/profile, roles: ROLE_USER }
- { path: ^/backhome, roles: ROLE_ADMIN }
- {
    path: ^/(food|category|virtue|user)/new,
    roles: ROLE_ADMIN,
    methods: [GET, POST],
}
- {
    path: ^/(food|category|virtue|user|recipe),
    roles: ROLE_ADMIN,
    methods: [GET, POST],
}
- {
    path: ^/(food|category|virtue|user)/\d+/edit,
    roles: ROLE_ADMIN,
    methods: [GET, POST],
}
- {
    path: ^/(food|category|virtue|user)/\d+,
    roles: ROLE_ADMIN,
    methods: POST,
}
- { path: ^/api/login, roles: PUBLIC_ACCESS }
- { path: ^/api/recipes, roles: ROLE_USER, methods: POST }
- { path: ^/api/recipes/\d+, roles: ROLE_USER, methods: PATCH }
- { path: ^/api/users, roles: PUBLIC_ACCESS, methods: POST }
- { path: ^/api/users/\d+, roles: ROLE_USER, methods: PATCH }
```

- **L'Environnement Symfony** est aussi un outil permettant d'améliorer la sécurité de notre application. En effet d'après le site <https://www.illy.fr/> « sous Symfony, le contrôle de la sécurité est simple et avancé, grâce à des fonctionnalités intégrées telles que la validation des données, la protection CSRF et la prévention des attaques XSS. Ce framework dispose de plusieurs options de sécurité tel que **SecurityBundle** qui fournit toutes les fonctionnalités d'authentification et d'autorisation nécessaires afin de sécuriser votre connexion et votre navigation. Il y a également **Doctrine** ou encore **Propel** pour interagir avec votre base de données. »

En effet en ce qui concerne l'ORM (Object-Relational Mapping) Doctrine il est intéressant d'évoquer que son utilisation peut être pertinente en termes de sécurité dans un projet Symfony. En outre il permet notamment :

- Une protection contre les attaques SQL Injection en utilisant les requêtes préparées.
- La validation des données lors de l'insertion ou de la mise à jour d'entités, ce qui permet de s'assurer que les données sont conformes aux règles de validation contribuant ainsi à éviter les erreurs et les vulnérabilités.
- La gestion des relations entre les entités en gérant les relations entre les entités de manière sécurisée, en prévenant les problèmes liés à la surcharge des clés étrangères, à la cascade des opérations, et à la gestion des entités liées.
- La gestion des transactions ce qui permet de garantir l'intégrité des données et d'éviter les problèmes de corruption des données.



- **Le système d'authentification** sous Symfony consiste à vérifier l'identité d'un utilisateur à l'aide d'informations d'identification sur un formulaire, à générer un token d'authentification, à stocker ce token et à vérifier sa validité pour permettre à l'utilisateur d'accéder aux ressources protégées de l'application.

Par exemple le code ci dessous codé en twig permet de définir un formulaire d'authentification qui s'affichera automatiquement lorsqu'un visiteur décidera de

se diriger sur l'une des différentes pages du backoffice. Cette fonctionnalité permet par consequent de sécuriser par un système d'identifiant et de mot de passe l'accès au backoffice.

```
{# templates/login/index.html.twig #}
{% extends 'base.html.twig' %}

{% block body %}
    {% if error %}
        {% condition qui empêche la connexion si une erreur est présente, sinon elle affiche un message d'erreur en utilisant la méthode 'trans' pour la traduction. #}
            <div>{{ error.messageKey|trans(error.messageData, 'security') }}</div>
        {% endif %}

        {% le formulaire est ici soumis à l'adresse 'app_login' en utilisant la méthode POST pour activer la connexion %}
        <form action="{{ path('app_login') }}" method="post">

            {% Champ de l'email %}
            <div class="login">
                <label for="username">Email:</label>
                <input type="text" id="username" name="_username" value="{{ last_username }}"/>
            </div>

            {% Champ du mot de passe %}
            <div class="login2" >
                <label for="password">Mot de passe:</label>
                <input type="password" id="password" name="_password"/>
            </div>

            {% Bouton de validation permettant la connexion %}
            <div class="loginButton">
                <button class="btn btn-success" type="submit">Connexion</button>
            </div>
        
```

- **Le hachage du mot** de passe est une composante importante de la sécurité de l'application notamment pour prévenir du vol de mot de passe par un visiteur malveillant.

Ainsi nous avons mis en place un système de hachage automatique de mot de passe lorsqu'un utilisateur l'édite lors de son inscription.

Pour ce faire les lignes de codes ci-dessous ont été implémenté dans le fichier `sécurité.yaml` afin de spécifier les paramétrages du hachage de mot de passe. Nous nous sommes servi de la documentation Symfony pour introduire ce code.



```
security:  
    password_hashers:  
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:  
            algorithm: auto  
            cost: 4 # Lowest possible value for bcrypt  
            time_cost: 3 # Lowest possible value for argon  
            memory_cost: 10 # Lowest possible value for argon
```

De plus, dans la fonction permettant de créer un utilisateur (extrait du UserController), il a fallu ajouter un algorithme permettant d'hacher le mot de passe au moment de la création d'un nouvel utilisateur.

Ci-dessous l'algorithme extrait de la fonction ‘create’ du UserController permettant de hasher le mot de passe:

```
// Algorithme permettant le Hash the password  
  
//Récupération du mot de passe de l'utilisateur au moment de sa création  
$plainPassword = $user->getPassword();  
  
//Utilisation de la fonction hashPassword qui renvoie le mot de passe hashé de l'utilisateur  
$hashedPassword = $passwordHasher->hashPassword($user, $plainPassword);  
  
//Modification et Stockage du MDP hashé dans la l'entité user  
$user->setPassword($hashedPassword);
```

## 2. Sécurité coté Front end

### LocalStorage et cookie

Le code ci-dessous, réalisé par l'équipe frontend et présent dans le fichier ‘userContext.js’ permet la création d'un contexte utilisateur (UserContext) avec React afin de gérer l'état de connexion d'un utilisateur.

L'état de connexion permet de déterminer si un utilisateur a été authentifié avec succès et a bien l'accès aux fonctionnalités réservées aux utilisateurs connectés, ou à l'inverse s'il doit être redirigé vers une page de connexion ou d'inscription. Pour stocker ces informations l'application utilise le localStorage qui est un espace de stockage local dans le navigateur web. Ainsi lorsqu'il s'agit de gérer l'état de connexion de l'utilisateur, il est possible de stocker des informations telles que le



token d'authentification, les données d'utilisateur (nom, email, etc.), ou d'autres informations nécessaires pour vérifier si un utilisateur est connecté ou non.



```
import React, { createContext, useEffect, useState } from "react";

export const UserContext = createContext();

const UserContextProvider = ({ children }) => {
// ici vous pouvez mettre ce que vous voulez votre user, votre token, etc...
  const [user, setUser] = useState(null);
  const [isLoggedIn, setIsLoggedIn] = useState(null);

  useEffect(() => {
    const storedUser = JSON.parse(localStorage.getItem("user"));
    if (storedUser) {
      setUser(storedUser);
      setIsLoggedIn(true);
    }
  }, []);

  const verifiedUser = () => {
    const storedUser = JSON.parse(localStorage.getItem("user"));
    return storedUser ? true : false;
  }

  const updateUser = (newUser) => {
    setUser(newUser);
    localStorage.setItem("user", JSON.stringify(newUser));
  };

  const logOut = () => {
    setUser(null);
    setIsLoggedIn(false);
    localStorage.removeItem("user");
  };

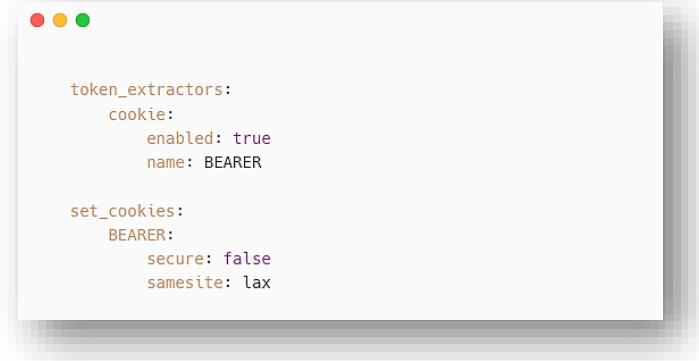
  return (
    <UserContext.Provider value={{ user, updateUser, isLoggedIn, setIsLoggedIn, logOut, verifiedUser }}>
      {children}
    </UserContext.Provider>
  );
};

export default UserContextProvider;
```

Cependant, Il est important de signaler que le localStorage est accessible uniquement depuis le même domaine et le même protocole (http ou https) que l'application qui y accède. Il est alors nécessaire de prendre en compte les questions de sécurité liées au stockage d'informations sensibles, telles que les tokens d'authentification. C'est pourquoi pour une gestion plus sécurisée de l'état de connexion de l'utilisateur, il nous a été recommandé d'utiliser les cookies côté backend en complément.



Or pour des questions de délais impartis nous n'avons pu configurer correctement l'utilisation des cookies sur la partie frontend malgré la configuration effective côté backend. Le code ci-dessous (non actif dans la version finale) présent dans le fichier 'lexik\_jwt\_authentication.yaml' permet de configurer l'extraction du JWT des requêtes entrantes et définit les cookies lors de la génération du JWT.



```
token_extractors:
  cookie:
    enabled: true
    name: BEARER

  set_cookies:
    BEARER:
      secure: false
      samesite: lax
```

Toujours d'après le site <https://yesweDev.bzh/>, voici quelques raisons pour lesquelles l'utilisation de cookies côté backend peut contribuer à renforcer la sécurité du code :

Ils permettent entre autre:

1. « Une protection contre les attaques XSS (Cross-Site Scripting) : Les attaques XSS sont des vulnérabilités courantes dans les applications web où des scripts malveillants sont injectés dans les pages web et exécutés côté client. En stockant les informations d'utilisateur sensibles dans les cookies sécurisés (marqués avec le drapeau HttpOnly), les données ne sont pas exposées aux attaques XSS, car elles ne sont pas accessibles via JavaScript.
2. Une protection contre les attaques CSRF (Cross-Site Request Forgery) : Les attaques CSRF sont des attaques où un site malveillant peut envoyer des requêtes HTTP au nom de l'utilisateur authentifié sur un autre site. En utilisant des cookies avec des jetons anti-CSRF, il est possible de protéger les requêtes sensibles contre de telles attaques, en vérifiant que les requêtes fournies du site légitime.
3. Une gestion sécurisée des informations d'utilisateur : Les cookies côté backend peuvent être chiffrés et signés pour garantir la confidentialité et l'intégrité des données de l'utilisateur. Cela évite la possibilité de manipulation ou d'accès non autorisé aux informations d'utilisateur.

4. Un contrôle fin des paramètres de cookie : Les cookies côté backend permettent un contrôle plus fin sur les paramètres de cookie, tels que la durée de vie, le domaine, le chemin et les drapeaux de sécurité. Cela permet de mieux gérer la sécurité et la confidentialité des données de l'utilisateur. »

En résumé les cookies sont des éléments clés de la gestion de l'état de session dans les applications web, et leur sécurité est un aspect important à prendre en compte pour protéger les données sensibles et garantir la confidentialité et l'intégrité des informations échangées entre le client et le serveur.

## VI. Gestion de projet

### A. Presentation de l'équipe et Explication des roles de chacun

L'équipe du projet GoodFood était composé de 5 membres, chacun ayant des rôles définis le premier jour du projet. Celle-ci se compose de la team :



**Gaël Cantonnnet** : Lead Dev Front et Référent Tech

**Aléxis Bizet** : Scrum Master et développeur frontend



**Abdoul Tambadou** : Référent Git et développeur backend

**Steven Organista** : Lead Dev Back / Référent Tech

**Dorian Guignard** : Product Owner et développeur backend

Gael et Alexis se sont occupés de la partie frontend du projet, après avoir suivi une spécialisation d'un mois sur React. De leur côté, Abdoul, Steven et moi-même étions responsables de la partie backend, après une spécialisation d'un mois sur le framework Symfony. Les rôles ont été correctement répartis dès le début en tenant compte des souhaits et des compétences de chacun.

En tant que Lead Developer et référent technique, Gael et Steven ont assuré la supervision du développement de l'application et ont apporté leur expertise technique aux autres membres de l'équipe en cas de besoin.

Alexis, en tant que Scrum Master, a organisé et programmé régulièrement les réunions de l'équipe et a consigné scrupuleusement les tâches réalisées par l'équipe dans le carnet de bord.

Abdoul, en tant que référent Git, a assuré son rôle en nous aidant à résoudre de nombreuses problématiques liées à la gestion des versions et aux conflits Git.

En tant que Product Owner, j'ai régulièrement réorienté le projet pour rester fidèle aux décisions initiales ou pour prendre les décisions finales concernant les aspects visuels ou l'évolution du concept du projet.

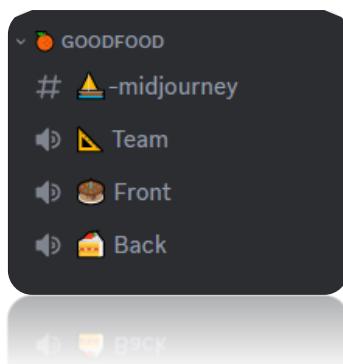


## B. Méthodologie de travail

En terme de méthodologie, nous avons appliqué la méthode Agile Scrum. Comme expliqué plus haut les rôles et les responsabilités de chaque membre de l'équipe étaient clairement définis, avec Gael et Steven en tant que Lead Developper et référents tech, Alexis en tant que Scrum Master, Abdoul en tant que référent Git, et moi-même en tant que Product Owner. Ainsi nous avons régulièrement organisé des réunions d'équipe, en général chaque matin à 9h00 afin d'orienter ce sur quoi nous allions travailler et redéfinir les besoins et les objectifs. Puis souvent vers 15h00 pour énumérer ce sur quoi chacun avait travaillé, ce qu'il restait à accomplir et les difficultés rencontrées.

De plus nous avons consigné les tâches réalisées et les difficultés rencontrés sur un carnet de bord (l'un personnel et l'autre d'équipe). Ainsi cette approche dite agile nous a permis de travailler de manière efficace et collaborative, en maximisant la communication et la collaboration entre nous cinq.

En terme d'organisation nous nous sommes répartis en deux équipes sur deux salons Discord la Team Front et la Team Back. Le troisième salon : 'Team' permettait de tous se réunir lors des réunions d'équipe ou de discuter avec un membre de l'école. Cf screenshot ci-dessous.



Nous avons été unanime pour exprimer que cette organisation avait été extrêmement pertinente pour développer un projet en équipe. Dans le sens où nous avions le sentiment de travailler dans des bureaux distincts mais qu'en cas de besoin nous pouvions intervenir sans hésiter vers l'un ou vers des salons pour une demande particulière. Ceci a permis de créer un climat très agréable pour chacun d'entre nous et facilitateur dans l'avancer respective de nos différentes fonctionnalités.

**Du côté back** nous avons fait le choix de travailler **conjointement et en temps réel** sur le projet dans le salon 'Back'. De manière général deux personnes du groupe travaillait sur une fonctionnalité et l'autre faisait des recherches ou bien travaillait sur

une autre fonctionnalité. Etant donné la très bonne entente entre nous 3, ce format d'organisation a particulièrement bien fonctionné. Il nous a notamment permis d'apprendre davantage via les compétences et les connaissances de chacun, de travailler de manière rapprochée dans un climat positif, d'argumenter en cas de désaccord, de se rassurer lorsque des difficultés importantes apparaissaient.

Néanmoins il est possible de citer quelques inconvénients concernant ce type d'organisation, notamment le fait d'être en communication toute la journée qui a pu engendrer à certains moments un effet de fatigue supplémentaire ou encore la nécessité de devoir argumenter souvent pour tester une idée.

Malgré ces inconvénients cette organisation a correctement fonctionné grâce à ses nombreux avantages et nous a permis de développer efficacement, tout en communiquant systématiquement en bonne intelligence.

## C. Les Outils utilisés

Nous avons utilisé plusieurs outils pour faciliter notre communication et notre travail d'équipe.

- Tout d'abord nous avons utilisé **Google Drive** pour échanger des documents partagés et Google Docs pour la conception du cahier des charges et la rédaction des carnets de bord.



- Pour la communication écrite ou les appels en cas de problème de connexion avec Discord, nous avons utilisé l'application **Slack**.



- **Discord** nous a permis de travailler sur trois salons distincts pour communiquer facilement à l'oral et partager nos écrans afin de pouvoir montrer nos progrès ou partager du code.



- Pour la création et le suivi des user stories, nous avons utilisé **Trello**.



- Nous avons utilisé **Visual Studio Code** comme environnement de développement intégré (IDE) pour travailler sur un code unique avec Live Share, ainsi que pour partager des serveurs et des terminaux.



- Enfin nous avons utilisé **Git** comme système de contrôle de version pour gérer nos modifications de code et opérer en équipe, tandis que **GitHub** a servi de plateforme pour héberger notre dépôt de code, faciliter les revues de code et notre travail de développement.



## D.Les sprints

Sur la période d'un mois nous avons réalisé 4 sprints pour notre projet:

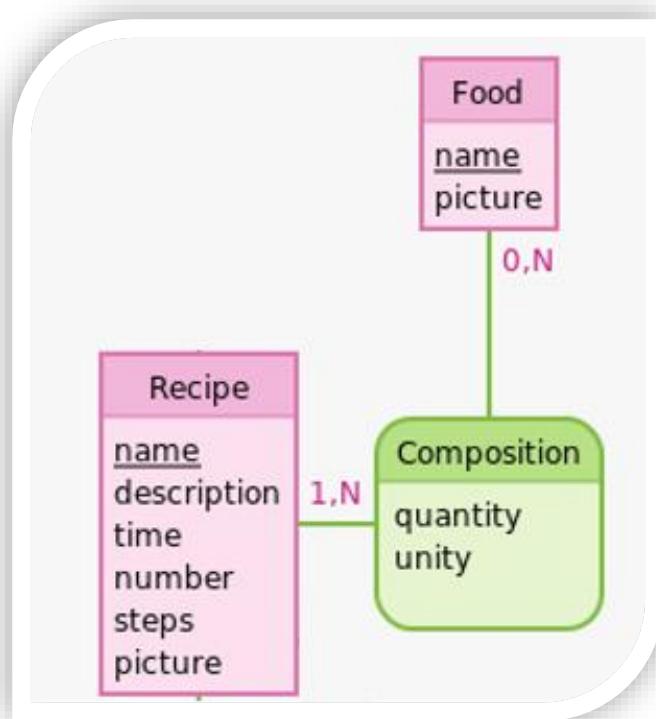
- ◆ Le **Sprint 0**, qui s'est déroulé du 13/02 au 20/02, a été confié à la conceptualisation du projet, notamment en rédigeant le cahier des charges. Cela a inclus la création du MCD (Modèle Conceptuel de Données), du MLD (Modèle Logique de Données), du dictionnaire de données, des wireframes, de la charte graphique et du logo, ainsi que la définition des noms de nos routes.
- ◆ Au cours du **Sprint 1**, qui s'est déroulé du 21/02 au 24/02, nous avons mis en place l'architecture du projet côté backend et frontend. Nous avons déterminé les fonctionnalités prioritaires, créé la base de données et recherché les données à implémenter. Nous avons également commencé à travailler sur le backoffice, créé l'API pour communiquer les données sur le dépôt frontend, et travaillé sur l'interface de la page d'accueil du FrontOffice. Du côté du Frontend, notre équipe s'est principalement concentrée sur la création des routes, la connexion d'un utilisateur, le responsive design et l'ajout d'autres fonctionnalités du côté de l'interface graphique.
- ◆ Pendant le **Sprint 2**, qui s'est déroulé du 27/02 au 03/03, notre attention s'est portée sur la sécurité côté backend, notamment en mettant en place les ACL, en utilisant le bundle JWT Token, en implémentant l'authentification d'un utilisateur et le hachage des mots de passe, ainsi qu'en réfléchissant aux étapes d'une recette et à leur intégration dans notre projet. Nous avons également résolu des bugs liés à l'API. Du côté du frontend, plusieurs fonctionnalités ont été développées, notamment le slider de la page d'accueil, la validation du mot de passe lors de la connexion, et l'affichage dynamique des données.
- ◆ Enfin, lors du **Sprint final** qui s'est déroulé du 06/03 au 10/03, nous nous sommes concentrés côté backend sur la validation des tests pour l'API et le backoffice, l'amélioration du design du backoffice, la fonctionnalité de téléchargement d'image, la configuration des cookies et jeu d'essai. Du côté du frontend, notre équipe s'est focalisée sur l'amélioration du design du site, la finalisation du formulaire d'ajout d'une recette, ainsi que la partie connexion et le profil d'un utilisateur.

## VII. Mes réalisations

### A. Sprint 0 : Elaboration de la table pivot ‘composition’ contenant les informations complémentaires ‘quantity’ et ‘unity’.

Lors de la réflexion concernant la création du MCD et du MLD nous nous sommes beaucoup attardé sur la cohérence de relations entre les différentes tables de notre BDD. Ainsi entre la table recette et la table ingrédient nous avons identifié le verbe correspondant à la relation nommé ‘composé de’. De ce fait et en raison des cardinalités 1N et 0N de chaque côté cela a entraîné la création d'une table pivot.

Cette table pivot a par logique été nommé ‘Composition’ et a permis de contenir deux informations complémentaires soit le nom de l'unité de mesure (mg, ml, L, kg, ... ) de l'ingrédient et la valeur de la quantité associée . En outre des unités variables en fonction de chaque recettes donc par conséquent non intégrable à la table ‘ingrédient’.



En utilisant le framework Symfony, j'ai créé une entité Composition qui est liée aux entités Food et Recipe. Voici ci-dessous les propriétés dans le code (situé dans le fichier Composition.php) qui permettent d'établir les relations entre ces trois entités:

```
class Composition
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     * @Groups({"compositions_get_collection", "compositions_get_item"})
     */
    private $id;

    /**
     * @ORM\ManyToOne(targetEntity=Recipe::class, inversedBy="compositions", cascade={"persist"})
     * @ORM\JoinColumn(nullable=false)
     */
    private $recipe;

    /**
     * @ORM\ManyToOne(targetEntity=Food::class, inversedBy="compositions", cascade={"persist"})
     * @Groups({"compositions_get_collection", "compositions_get_item",
     *          "recipes_get_collection", "recipes_get_item"})
     */
    private $food;

    /**
     * @ORM\Column(type="string", length=20, nullable=true)
     * @Groups({"compositions_get_collection", "compositions_get_item",
     *          "recipes_get_collection", "recipes_get_item"})
     */
    private $unity;

    /**
     * @ORM\Column(type="string", nullable=true)
     * @Groups({"compositions_get_collection", "compositions_get_item",
     *          "recipes_get_collection", "recipes_get_item"})
     */
    private $quantity;
```

Les annotations de propriétés sont expliquées ci-dessous :

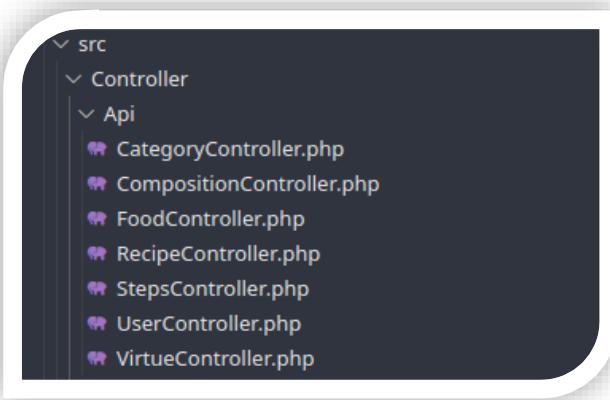
- Pour la propriété \$recipe et la propriété \$food l'annotations **Many-to-One** indiquent qu'il y a une relation avec l'entité Recipe
- "**inversedBy**" signifie que la propriété "compositions" dans l'entité Recipe est utilisée comme propriété inverse de cette relation.
- L'option "**cascade={"persist"}**" signifie que lorsque l'on a besoin de persister une entité Composition, les entités Recipe et Food associées seront également persistées automatiquement.
- L'annotation "**@ORM\JoinColumn(nullable=false)**" indique que cette relation est obligatoire et donc non nullable .
- L'annotation **@Groups** utilisée sur la propriété \$food incluant les groupes "compositions\_get\_collection", "compositions\_get\_item", "recipes\_get\_collection", et "recipes\_get\_item" permet d'inclure la propriété lors de la sérialisation en objet Json.

Pour conclure, ces différentes annotations attachées aux propriétés \$food et \$recipe de l'entité Composition vont principalement permettre de paramétriser la sérialisation et la désérialisation de ces propriétés en contrôlant quels attributs seront inclus ou exclus lors de la conversion des objets en format JSON lors de requêtes émises ou reçues par l'API.

## B. Sprint 1 : Durant ce sprint j'ai principalement contribué à la mise en place de l'API et au test des requêtes avec le logiciel Insomnia.

Dans un premier temps j'ai conçu les controller apparentés aux différentes entités. Le contrôleur d'une API, dans le contexte du framework Symfony , va permettre de gérer les requêtes HTTP entrantes et les traiter pour produire une réponse appropriée à renvoyer au client. Le contrôleur va donc agir comme un intermédiaire entre le client, qui envoie les requêtes, et les entités du projet, qui représente les données de l'application.

*Ci-dessous la liste des contrôleurs de l'API :*



Dans les différents controller j'ai ainsi pu définir et gérer les routes de l'API, c'est-à-dire les URL et les méthodes HTTP (comme GET, POST, PUT, DELETE) auxquelles l'API devra répondre.

Le code commenté ci-dessous présente deux fonctions que j'ai produit situé dans du controller FoodController, la première permettant de recuperer la liste de tous les ingredients et la seconde de modifier un ingrédient contenu dans la BDD.

A titre informatif ces fonctions présentes dans le controller sont composées de trois segments. Le premier segment est la partie annotation définissant le nom générique de la route, le nom de l'url qui sera employé par les utilisateurs de l'API et la méthode utilisé, GET en l'occurrence.

```
/**  
 * @Route("/api/foods", name="app_api_foods", methods={"GET"})  
 */
```

Le deuxième segment de ce code concerne le nom de la fonction et ses paramètres, incluant ici le paramètre **\$foodRepository** qui est une injection de dépendance. Cette technique permet d'implémenter un objet dans la fonction sans avoir besoin de le créer entièrement, mais simplement pour utiliser certains éléments de sa classe



comme une fonction ou une propriété. L'utilisation de 'JsonResponse' permet quant à lui de générer une réponse HTTP au format JSON.

```
/*
public function index(FoodRepository $foodRepository): JsonResponse
{
```

Le troisième segment correspond à l'ensemble des instructions qui seront exécuté par la fonction lorsque celle-ci sera appelée. Ainsi voici la fonction 'index' commenté ci-dessous :

```
/**
 * @Route("/api/foods", name="app_api_foods", methods={"GET"})
 */
public function index(FoodRepository $foodRepository): JsonResponse
{
    // "FoodRepository" permet de récupérer tous les ingredients à partir de la base
    // de données . Les résultats sont stockés dans une variable appelée "$foods".
    $foods = $foodRepository->findAll();

    // permet de générée une réponse JSON à renvoyer au frontend. Elle utilise la
    // méthode "json" du contrôleur pour transformer les données au format JSON. Cette
    // fonction native 'json' doit comprendre 4 arguments.
    return $this->json([
        // On précise à la méthode 'json' les données à inclure dans la reponse
        // soit un tableau associatif contenant les aliments récupérés à partir du
        // repository
        ['foods' => $foods],
        // Ici on spécifie le code de statut HTTP 200 pour indiquer que la requête
        // s'est déroulée avec succès.
        Response::HTTP_OK,
        // On indique ici les en-têtes HTTP de la réponse. Ici, aucun en-tête n'est
        // spécifié.
        [],
        // Ici, on identifie le groupe "foods_get_collection" qui est utilisé pour
        // déterminer quels attributs de l'entité "Food" doivent être inclus dans la
        // réponse JSON
        ['groups' => "foods_get_collection"]
    ]);
}
```



La seconde fonction nommée ‘patch’ du controller FoodController permet le bon fonctionnement de la mise à jour/modification (PATCH) d'un ingrédient dans l'API. Cette méthode comprend plusieurs étapes définie plus précisément dans les commentaires ci-dessous

```
/*
 * Update food
 *
 * @Route("/api/foods/{id<\d+>}", name="app_api_patch_foods_item", method={"PATCH"})
 */
public function patch(Food $food = null, Request $request, SerializerInterface
$serializer, ValidatorInterface $validator, EntityManagerInterface $entityManager)
{
    //Si un ingredient n'a pas été trouvé dans la BDD la requete retournera un message
    //d'erreur : ingredient non trouvé
    if ($food === null) {
        return $this->json(['message' => 'ingrédient non trouvé.'],
        Response::HTTP_NOT_FOUND);
    }

    // Sinon le code récupere le contenu de la requête et execute le décodage du format
    // JSON
    $jsonContent = json_decode($request->getContent(), true);

    // Les propriétés de l'objet $food sont ensuite mise à jour des avec les données de
    // la requête via l'utilisation des setter de l'entité Food.
    $patchFood = $food
        ->setName($jsonContent['name'])
        ->setNameImage($jsonContent['nameImage']);

    // Les nouvelles données sont ensuite enregistrées et envoyées à la base de données
    // via l'entityManager (sorte de super Repository)
    $entityManager->persist($patchFood);
    $entityManager->flush();

    // On retourne enfin la réponse en JSON avec les données de l'objet $food mis à jour
    // afin de les communiquer via l'API
    return $this->json(
        ['food' => $patchFood],
        Response::HTTP_OK,
        [],
        ['groups' => 'foods_get_item']
    );
}
```



Enfin, afin de pouvoir tester ces différentes méthodes j'ai utilisé le logiciel de requête Insomnia.

Ci-dessous une liste non exhaustive des différents tests effectués sur les routes de l'API côté backend.

The screenshot shows the Insomnia REST Client interface with the title "Dashboard / GoodFood". The main area displays a list of API endpoints categorized by method:

- POST**: POST imageUser, POST Add ImageToRecipe, POST recipe
- DEL**: DELETE Recipe
- GET**: GET ConnectUser, GET Recipe, GET Login, GET User, GET Category, GET One Category, GET All Category, POST Composition, PATCH Composition, GET One Composition
- PTCH**: PATCH Category, PATCH Composition
- GET**: PB Requête API POST Users

At the top of the interface, there are buttons for "No Environment" and "Cookies", a "Filter" input field, and a set of navigation icons.

C. Pendant le **Sprint 2**, j'ai réalisé une vue du backoffice pour l'entité "vertu" et ses formulaires associés en utilisant l'architecture MVC.

Les étapes incluent la création des entités pour concevoir la partie Modèle, la mise en place des routes et des méthodes associées dans les contrôleurs, la création et la configuration des formulaires pour les méthodes PATCH et POST, par exemple, le formulaire d'ajout d'une recette ou une modification d'un ingrédient. Enfin, nous avons conçu les vues qui représentent la partie visuelle de l'application avec laquelle l'administrateur peut interagir, notamment l'interface du backoffice.

La vue pour l'entité "vertu" a été développée en utilisant les langages Twig et HTML. En termes de structure, le fichier base.html.twig contient les balises head avec les styles, le header avec le menu de navigation, le footer avec les boutons de déconnexion, de retour à la page d'accueil ou de retour sur le frontOffice . Il contient également le bloc Twig body `{% block body %} <main> </main> {% endblock %}` qui permet d'inclure des fragments de template à l'intérieur de ce bloc. Ces fragments de template sont des fichiers distants qui seront développés indépendamment mais en reprenant toujours la même base issu du fichier 'base.html.twig'.

*Le fichier page suivante montre l'intégralité de la structure du fichier 'base.html.twig', codé en language html et via le moteur de rendu de template twig.*

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>
            {% block title %}
                GoodFood Backoffice
            {% endblock %}
        </title>
        <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 128 128%22><text y=%221.2em%22 font-size=%22296%22></text></svg>">
        {# Run `composer require symfony/webpack-encore-bundle` to start using Symfony UX #}
        {% block stylesheets %}
            {{ encore_entry_link_tags('app') }}
            <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-GlhltQ8lRABdZLl603oVMWSktQOp6b7In1Zl3/Jr59b6EGGoIlaFkw7cmDA6j6gD" crossorigin="anonymous">
            <link rel="stylesheet" href="../css/styles.css">
        {% endblock %}
        {% block javascripts %}
            {{ encore_entry_script_tags('app') }}
            <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-w6AqPfDkMBDXo30jS1Sgez6prx3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTF7CXvN" crossorigin="anonymous"></script>
        {% endblock %}
    </head>
    <body>
        <header>

            <nav>
                <ul class="nav nav-pills nav-justified">
                    <li class="nav-item">
                        <a class="nav-link " aria-current="page" href="{{ path('app_food_index') }}>
                            Ingrédients
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link " href="{{ path('app_recipe_index') }}>
                            Recettes
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ path('app_virtue_index') }}>
                            Vertus
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ path('app_category_index') }}>
                            Categories
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ path('app_user_index') }}>
                            Utilisateurs
                        </a>
                    </li>
                </ul>
            </nav>
        </header>
        {% block body %}<main></main>{% endblock %}

        <footer>
            <nav>

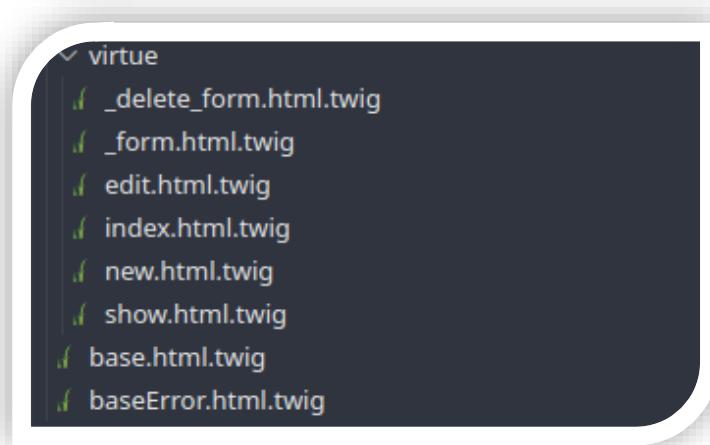
                <ul class="nav justify-content-center">
                    <li class="nav-item">
                        <a class="nav-link " aria-current="page" href="{{ path('app_backhome') }}>
                            Home
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="http://localhost:3000/">GoodFood</a>
                    </li>
                    <li class="nav-item">
                        <a class="btn btn-danger" href= "{{ path('app_logout') }}>
                            Déconnexion
                        </a>
                    </li>
                </ul>
            </nav>
        </footer>
    </body>
</html>

```

Ce document montre comment est articulé l'arborescence des fichiers de vue sous twig concernant l'entité Virtue.

Les fichiers représenté ici permettent de bas en haut

- d'afficher une page pour visualiser une vertu sélectionnée
- d'afficher une page listant toutes les vertus
- d'afficher un formulaire de modification
- d'afficher un formulaire d'ajout
- de générer un formulaire Symfony
- d'ajouter un bouton de suppression de donnée



- Aperçu de la page index des vertus du backoffice

Index des vertus					
<b>Id</b>	<b>Nom</b>	<b>Description</b>	<b>Image</b>	<b>actions</b>	
1	Vitamines	Le corps ne fabrique pas de vitamines, à l'exception de la vitamine D. Nous devons nous en procurer dans nos aliments.		<a href="#">Détail</a>	<a href="#">Edition</a>
2	Rajeunissant	Il existe effectivement des aliments qui permettent de ralentir le vieillissement. Ce sont tous ceux qui sont riches en antioxydants (vitamine C, betacarotène, sélénium...). On trouve ces composés notamment dans les fruits et légumes. Ainsi, la vitamine C se trouve dans les agrumes, les fraises, les kiwis. La vitamine E se trouve dans le germe de blé... Pour faire le plein de caroténoïdes, ce n'est pas compliqué, il faut choisir les fruits et légumes les plus colorés : orange, rouge mais aussi vert. L'effet de ces antioxydants est notamment visible au niveau de la peau. D'ailleurs, il suffit de voir le teint d'un fumeur, dont le statut en antioxydants est bas...		<a href="#">Détail</a>	<a href="#">Edition</a>
3	Détox	Après les excès des fêtes, on est bien décidé à faire du bien à notre corps en mettant le holà sur les graisses et les plats trop riches. Pour éliminer les toxines, réduire les troubles de la digestion, retrouver du tonus et une belle peau, sans se ruiner, on vous guide pour remplir votre panier de marché d'aliments détox de saison, le tout accompagné de conseils et d'idées recettes, évidemment.		<a href="#">Détail</a>	<a href="#">Edition</a>
4	Boost	En moyenne, les Français dorment 6 h 55 par jour, sieste comprise, soit bien moins que ce dont ils estiment avoir besoin... Résultat ? La fatigue peut s'installer et avec elle, le manque d'énergie. Alors plutôt que de vous jeter sur votre troisième tasse de café de la matinée, misez sur des aliments riches en		<a href="#">Détail</a>	<a href="#">Edition</a>

- Aperçu du formulaire d'ajout d'une vertu

The screenshot shows a web interface for adding a virtue. At the top, there is a navigation bar with tabs: Ingrédients, Recettes, Vertus, Categories, and Utilisateurs. The 'Vertus' tab is active. Below the navigation bar, the page title is 'Ajouter une vertu'. There are three input fields: 'Name' (with an empty input box), 'Description' (with an empty input box), and 'Image (JPG, PNG, JPEG, GIF file)' (with a file input field containing 'Aucun fichier choisi' and a 'Choisir un fichier' button). Below these fields are two buttons: a yellow 'Save' button and a blue 'Retour à l'index' button. At the bottom of the page is a green footer bar with links: Home, GoodFood (highlighted in red), and Déconnexion.

Pour que ce formulaire soit opérationnel, il est nécessaire d'utiliser plusieurs fichiers qui sont complémentaires et connectés les uns aux autres. Tout d'abord, le code va utiliser le fichier de **type de formulaire (image1)** pour définir la structure de base du formulaire et les champs qui le composent. Cette structure est ensuite utilisée par le fichier **\_form.html.twig(image2)** pour générer dynamiquement le formulaire avec les champs correspondants. Enfin, toutes ces informations sont encapsulées dans le fichier **new.html.twig(image3)**, qui est responsable de l'affichage de la page complète contenant le formulaire et tous les autres éléments nécessaires.

## (Image1 : code commentée de la classe VirtueType)

```
class VirtueType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        // $builder permet d'ajouter les champs au formulaire
        $builder
            // le champ 'name' de la vertu
            ->add('name')

            // le champ 'description' de la vertu
            ->add('description')

            // le champ 'nameImage' correspondant à la propriété de l'entité Virtue en charge du téléchargement d'image
            ->add('nameImage', FileType::class, [
                // Ce champ dispose de plusieurs paramètres :

                // le label qui sera indiqué au dessus du champ
                'label' => 'Image (JPG, PNG, JPEG, GIF file)',

                // L'option 'mapped' qui est définie sur false,
                // ce qui signifie que ce champ ne sera pas mappé
                'mapped' => false,

                // L'option 'required' qui est définie sur false qui signifie que ce champ n'est pas obligatoire.
                'required' => false,

                // Les contraintes qui vérifie si l'utilisateur télécharge le type de fichier attendu par l'application.
                'constraints' => [
                    new FileConstraint([
                        'maxSize' => '1024k',
                        'mimeTypes' => [
                            'image/jpeg',
                            'image/png',
                            'image/jpg',
                            'image/gif',
                        ],
                        // Enfin ceci correspond au message qui sera envoyé si les contraintes ne sont pas respectées
                        'mimeTypesMessage' => 'Please upload a valid JPG, PNG, JPEG or GIF image',
                    ])
                ],
            ]);
    }

    // La méthode suivante permet de définir les options par défaut du formulaire en se basant sur l'entité virtue pour la soumission et la validation du formulaire.
    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Virtue::class,
        ])
    }
}
```

## (Image2 : code commentée du fichier *\_form.html.twig*)

```
● ● ●

<!-- Ceci génère la balise d'ouverture du formulaire -->
{{ form_start(form) }}

<!-- Ici on affiche les widgets de chaque champ du formulaire, et on génère les balises HTML pour les champs -->
{{ form_widget(form) }}

<!-- Création du bouton de formulaire avec une classe "btn btn-warning" -->
<button class="btn btn-warning">{{ button_label|default('Enregistrer') }}</button>

<!-- Ceci génère la balise de fermeture du formulaire -->
{{ form_end(form) }}
```

## (Image3 : code issu du fichier *new.html.twig*)

```
● ● ●

{% extends 'base.html.twig' %}

{% block title %}New Virtue{% endblock %}

{% block body %}


# Ajouter une vertu


{{ include('virtue/_form.html.twig') }}

Retour à l'index


{% endblock %}
```

D. Pendant le **Sprint final**, j'ai implémenté la fonctionnalité qui permet à un utilisateur de télécharger une image via l'application.

Cette fonctionnalité permet entre autre d'ajouter un avatar pour sa photo de profil, de créer une photo correspondante à une nouvelle recette via le formulaire du FrontOffice, ou de modifier une image correspondante à une vertu depuis le BackOffice. Initialement, j'ai préféré utiliser le bundle Symfony VichUploader Bundle pour simplifier cette fonctionnalité. Cependant, après avoir passé beaucoup de temps à essayer de configurer les paramètres et à comprendre le fonctionnement de ce composant, j'ai décidé de créer la fonctionnalité à partir de zéro en suivant la documentation Symfony et notamment la page [https://symfony.com/doc/current/controller/upload\\_file.html](https://symfony.com/doc/current/controller/upload_file.html).

En outre le développement de cette fonctionnalité m'a demandé beaucoup de temps et de recherche de par sa complexité mais m'a apporté une certaine satisfaction après l'avoir finalisé. J'ai malgré tout eu besoin d'aide de notre tutrice Adélie Castel notamment pour l'articulation de l'algorithme permettant le renommage de l'image.

*Ci-dessous je présente de manière détaillé et commenté la fonction 'new' extraite du CategoryController pour la partie backOffice.*



```

/**
 * @Route("/new", name="app_category_new", methods={"GET", "POST"})
 */
public function new(Request $request, CategoryRepository $categoryRepository, SluggerInterface $slugger):
Response
{
    $category = new Category();
    // Crée une nouvelle instance de la classe Category

    $form = $this->createForm(CategoryType::class, $category);
    // Crée un formulaire en utilisant la classe CategoryType et l'objet $category comme donnée

    $form->handleRequest($request);
    // Gère la requête HTTP pour le formulaire

    if ($form->isSubmitted() && $form->isValid()) {
        // Vérifie si le formulaire a été soumis et est valide
        $categoryRepository->add($category, true);
        // Ajoute l'objet $category à la base de données via le CategoryRepository avec une option de flush
        // (sauvegarde) activée

        /** @var UploadedFile $imageFile */
        $imageFile = $form->get('nameImage')->getData();
        // Récupère le fichier uploadé depuis le champ 'nameImage' du formulaire

        if ($imageFile) {
            // Vérifie si un fichier a été uploadé

            $originalFilename = pathinfo($imageFile->getClientOriginalName(), PATHINFO_FILENAME);
            // Récupère le nom original du fichier

            $safeFilename = $slugger->slug($originalFilename);
            // Génère un nom de fichier sûr en utilisant le SluggerInterface
            $newFilename = $safeFilename . '-' . uniqid() . '.' . $imageFile->guessExtension();
            // Génère un nouveau nom de fichier unique en ajoutant un identifiant unique et l'extension du
            // fichier d'origine

            try {
                $imageFile->move($this->getParameter('categoryPic_directory'), $newFilename);
                // Déplace le fichier uploadé vers le répertoire de destination avec le nouveau nom de fichier

            } catch (FileNotFoundException $e) {
                $this->addFlash('error', 'Votre téléchargement a échoué');
                // Ajoute un message flash d'erreur
                return $this->redirectToRoute('app_category_new');
                // Redirige vers la page de création du formulaire
            }

            $category->setNameImage('images/categoryPic/' . $newFilename);
            // Met à jour le nom du fichier dans l'objet $category
        }

        $categoryRepository->add($category, true);
        // Ajoute à nouveau l'objet $category à la base de données avec la mise à jour du nom du fichier
    }

    return $this->redirectToRoute('app_category_index', [], Response::HTTP_SEE_OTHER);
    // Redirige vers la page d'index des catégories avec un code de statut HTTP 303 SEE OTHER
}

return $this->renderForm('category/new.html.twig', [
    // Rend la vue du formulaire de création de catégorie
    'category' => $category,
    // Passe l'objet $category à la vue
    'form' => $form,
    // Passe le formulaire $form à la vue
]);
}

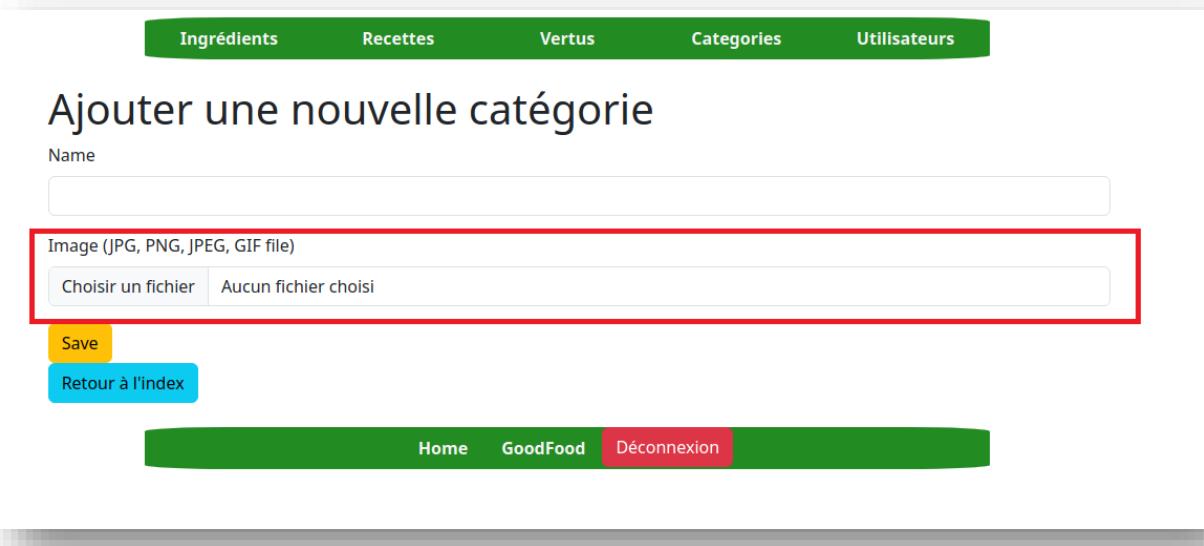
```

Ci-dessous sont présentés les paramètres de configuration issus du fichier services.yaml. Ces paramètres permettent d'indiquer le chemin directionnelle que doivent emprunter les images téléchargées par un utilisateur.



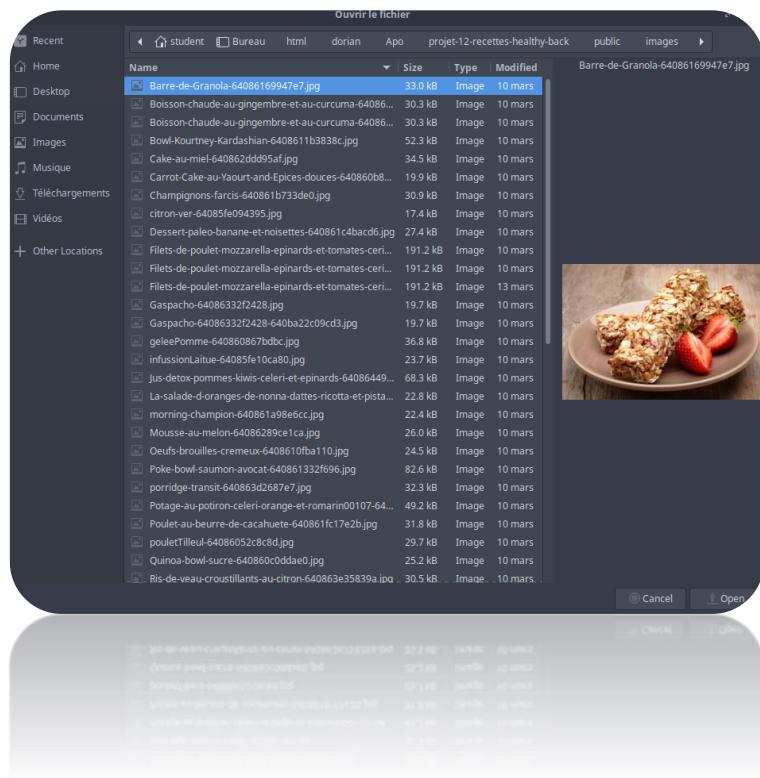
```
parameters:
  recipePic_directory: "%kernel.project_dir%/public/images/recipePic"
  foodPic_directory: "%kernel.project_dir%/public/images/foodPic"
  categoryPic_directory: "%kernel.project_dir%/public/images/categoryPic"
  virtuePic_directory: "%kernel.project_dir%/public/images/virtuePic"
  userPic_directory: "%kernel.project_dir%/public/images/userPic"
```

Ci-dessous est représenté l'interface graphique du backoffice permettant d'ajouter une catégorie et de télécharger une image associée choisie en local.



The screenshot shows a web-based administration interface for managing categories. At the top, there is a navigation bar with tabs: Ingrédients, Recettes, Vertus, Categories (which is the active tab), and Utilisateurs. Below the navigation bar, the title "Ajouter une nouvelle catégorie" (Add new category) is displayed. The main form has two input fields: "Name" and "Image (JPG, PNG, JPEG, GIF file)". The "Image" field is highlighted with a red border, indicating it is the current focus or the next step. Below the image field are two buttons: "Choisir un fichier" (Select file) and "Aucun fichier choisi" (No file selected). At the bottom of the form are two buttons: "Save" (yellow background) and "Retour à l'index" (blue background). At the very bottom of the page is a footer navigation bar with links: Home, GoodFood (the logo, which is a stylized green leaf with "GF" in the center), and Déconnexion (Logout).

Ci-dessous l'interface Linux apparaissant après avoir cliqué sur le bouton ‘choisir un fichier’ permettant de rechercher un fichier et dans ce cas précis le fichier image.



Voici le résultat obtenu après que l'utilisateur ai cliqué sur le bouton '**'open'**'. Cela inscrira le nom de l'image dans le champ Image qui sera ensuite transmis à l'application pour la récupération, la duplication et eventuellement la modification du fichier.

Ingrédients	Recettes	Vertus	Categories	Utilisateurs
<h3>Ajouter une nouvelle catégorie</h3>				
Name				
Granola				
Image (JPG, PNG, JPEG, GIF file)				
Choisir un fichier	Barre-de-Granola-64086169947e7.jpg			
<b>Save</b>				
<a href="#">Retour à l'index</a>				
<a href="#">Home</a> <a href="#">GoodFood</a> <a href="#">Déconnexion</a>				

## VIII. Presentation du jeu d'essai

Comme expliqué précédemment, dans le cadre de notre projet, nous avons développé une API permettant d'envoyer et de recevoir des requêtes pour créer une recette via un formulaire côté FrontOffice. Ceci ayant pour objectif de pouvoir partager sa propre recette à travers l'application GoodFood.

La création de cette API nous a demandé un temps de travail important et la résolution de nombreux problèmes pour assurer son bon fonctionnement et garantir la validité des données dans la base de données.

Ainsi cette fonctionnalité comprend plusieurs étapes pour être opérationnelle :

La première partie concerne le développement d'un formulaire d'ajout sur le FrontOffice pour permettre aux utilisateurs de saisir les informations nécessaires à la création d'une recette.

Arborescence des fichiers JS et CSS côté Frontend ayant permis le développement de l'interface du formulaire.

```
▽ components
  ▽ AddRecipe
    ▽ AddRecipeFinish
      # AddRecipeFinish.css
      JS AddRecipeFinish.js
    ▽ AddRecipeFoods
      # AddRecipeFoods.css
      JS AddRecipeFoods.js
    ▽ AddRecipeInfo
      JS AddRecipeInfo.js
    ▽ AddRecipeSteps
      JS AddRecipeSteps.js
      # AddRecipe.css
      JS AddRecipe.js
```

Rendu du formulaire sur la page d'ajout pour un utilisateur connecté

### Ajouter une recette

Ma recette      Mes ingrédients      Ma préparation      Validation

\* Titre de la recette:

Description rapide de la recette:

Temps avant de se régaler (préparation + cuisson) en minute:

Temps de cuisson en minute:

Temps de préparation en minute:

Nombre de convives:

Catégorie:

Vetue:

**Validez**

Envoi des données à l'API via Axios côté FrontEnd sur le serveur actif backend port 8080:



```
//Code extrait du fichier AddRecipeFoods.js
const fetchResults = () => {
  setLoading(true);
  axios.get(`http://0.0.0.0:8080/api/foods`)
    .then((response) => {

      setFoods(response.data[0].foods);
    })
    .catch((error) => {
      console.error(error);
    })
    .finally(() => {
      setLoading(false);
    });
};
```



Envoi des données à la BDD. Pour ce faire, utilisation des fonctions de l'entityManager persist et flush sur l'entité 'recipe' via la fonction 'create' du RecipeController de l'API coté BackEnd:

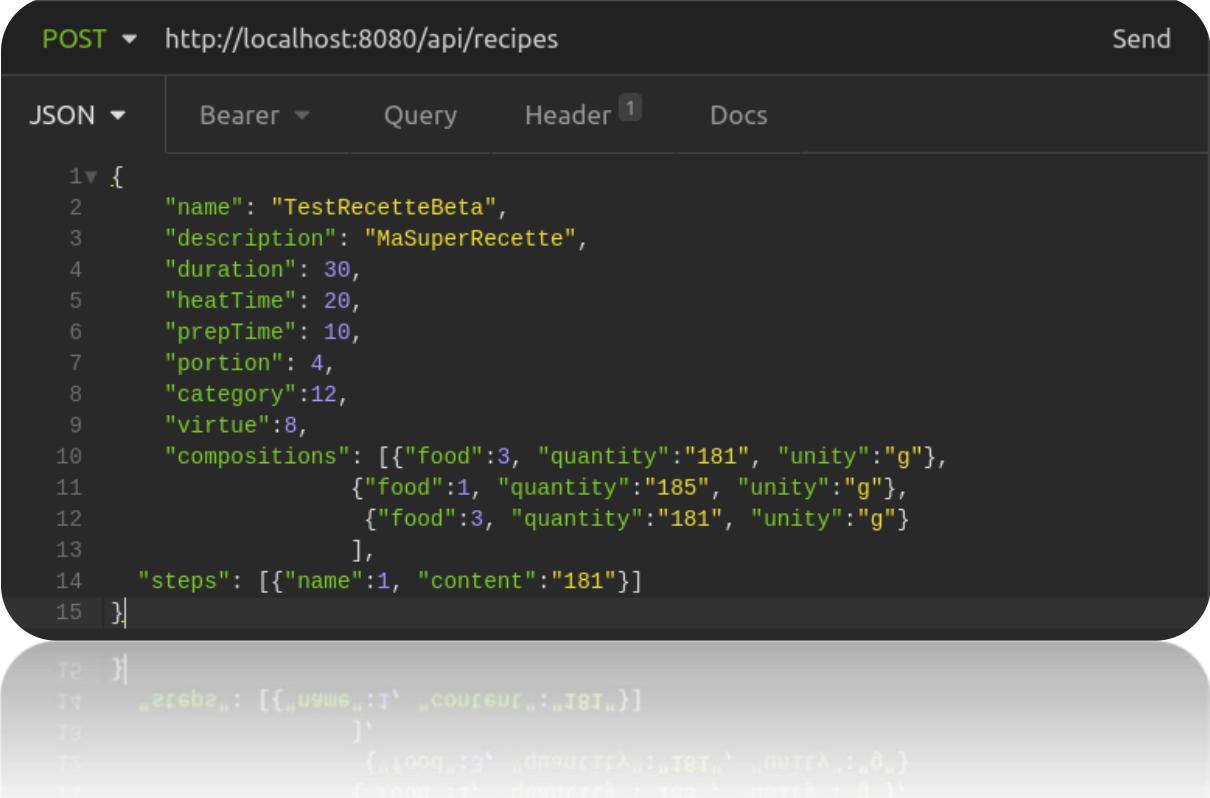
```
// Persist et flush  
$entityManager->persist($recipe);  
$entityManager->flush();
```

Pour conclure, plusieurs étapes sont donc nécessaires pour créer une recette sur le FrontOffice et tester la route API, notamment la création d'un formulaire, la validation des données, l'envoi des données à l'API, la gestion des erreurs, et les tests et débogage pour assurer un fonctionnement correct.

Aussi, pour tester l'ensemble des routes de l'API, nous avons donc **mis en place un jeu d'essai** en utilisant le logiciel d'exécution de requêtes **Insomnia**. Cela nous a permis de vérifier les fonctionnalités de l'API, de s'assurer que les requêtes étaient correctement traitées et que les données étaient enregistrées de manière appropriées dans la base de données.

→ Explication des étapes sur les pages ci-dessous concernant la route API de création d'une recette :

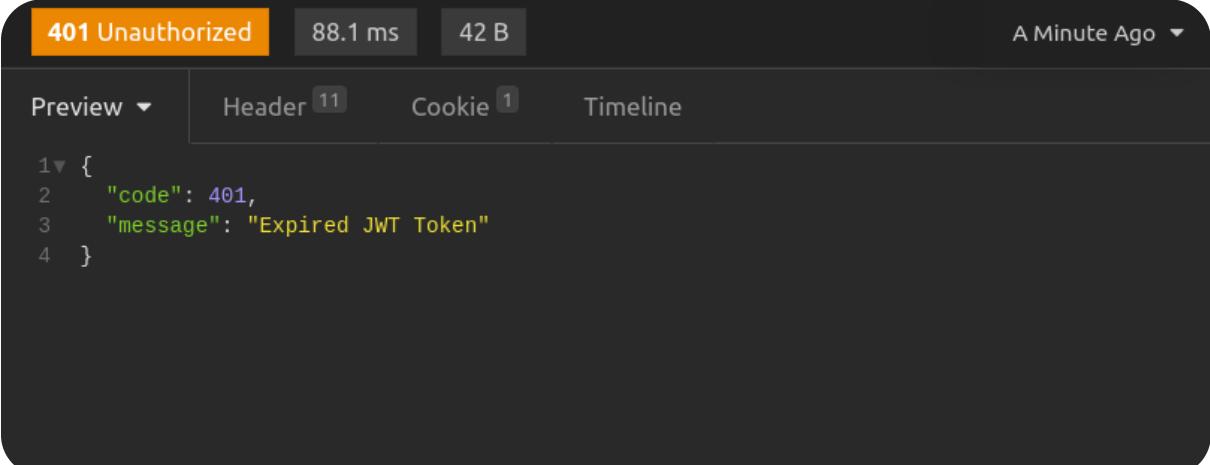
- Requete via la route API « api/recipes » en methode post pour l'ajout d'une recette dans la BDD.



The screenshot shows a POST request to `http://localhost:8080/api/recipes`. The JSON body contains the following recipe data:

```
1▼ {
2  "name": "TestRecetteBeta",
3  "description": "MaSuperRecette",
4  "duration": 30,
5  "heatTime": 20,
6  "prepTime": 10,
7  "portion": 4,
8  "category": 12,
9  "virtue": 8,
10 "compositions": [{"food": 3, "quantity": "181", "unity": "g"}, {"food": 1, "quantity": "185", "unity": "g"}, {"food": 3, "quantity": "181", "unity": "g"}],
11 ],
12 "steps": [{"name": 1, "content": "181"}]
13 }
14 }
```

- Erreur 401 lié à l'expiration du token, par conséquent génération d'un nouveau token qui sera ensuite injecté dans le bearer de la requete



The screenshot shows a 401 Unauthorized response. The JSON body contains the following error message:

```
1▼ {
2  "code": 401,
3  "message": "Expired JWT Token"
4 }
```

- Envoi de la requête, réponse 201 obtenu indiquant que la requête a réussi et qu'une ressource a bien été créée.

201 Created    221 ms    711 B    Just Now

Preview ▾    Header 11    Cookie 1    Timeline

```

1▼ {
2▼   "recipe": {
3      "id": 90,
4      "name": "TestRecetteBeta",
5      "description": "MaSuperRecette",
6      "duration": 30,
7      "heatTime": 20,
8      "prepTime": 10,
9      "portion": 4,
10     "compositions": [
11       {
12         "food": {
13           "id": 3,
14           "name": "Tisane(s) de menthe poivrée",
15           "nameImage": "images\\foodPic\\tisaneMenthePoivre-64088016c554c.jpg"
16         },
17         "unity": "g",
18         "quantity": "181"
19       },
20       {
21         "food": {
22           "id": 1,
23           "name": "Laitue(s)",
24           "nameImage": "images\\foodPic\\laitue-640876c58c422.jpg"
25         },
26         "unity": "g",
27         "quantity": "185"
28       },
29       {
30         "food": {
31           "id": 3,
32           "name": "Tisane(s) de menthe poivrée",
33           "nameImage": "images\\foodPic\\tisaneMenthePoivre-64088016c554c.jpg"
34           "unity": "g"
35           "quantity": "181"
36         },
37         "food": {
38           "id": 3,
39           "name": "Tisane(s) de menthe poivrée",
40           "nameImage": "images\\foodPic\\tisaneMenthePoivre-64088016c554c.jpg"
41           "unity": "g"
42         }
43       }
44     ]
45   }
46 }
```



- Vérification du résultat dans la table ‘recipe’ de la BDD ‘goodfood’:

<input type="checkbox"/>	edit	27	Crème brûlée au citron, coulis d'abricots et crème vanillée	NULL	45	35	10	6	6	11	
<input type="checkbox"/>	edit	28	Tarte aux pruneaux et aux abricots secs	NULL	20	0	5	4	6	9	
<input type="checkbox"/>	edit	29	Gaspacho	NULL	10	0	5	2	6	12	
<input type="checkbox"/>	edit	30	Smoothie vert	NULL	75	50	25	2	6	10	
<input type="checkbox"/>	edit	31	Riz complet aux légumes	NULL	10	0	5	1	6	13	
<input type="checkbox"/>	edit	32	Porridge Transit	NULL	15	0	10	4	1	9	
<input type="checkbox"/>	edit	33	Tartare de betterave à la truite fumée	A la recherche d'une entrée rapide à réaliser pour recevoir ? Testez ce tartare tout en fraîcheur et...	105	45	60	4	1	10	
<input type="checkbox"/>	edit	34	Ris de veau croustillants au citron	En panne d'idées pour cuisiner les ris de veau? Essayez cette recette simplissime à préparer qui ne...	10	0	10	4	1	12	
<input type="checkbox"/>	edit	35	Jus vitaminé pommes, kiwis, céleri et épinards	Cette recette de boisson vitaminé est signée Anne Lataillade du blog papillesetpupilles.fr. Réalisé...	20	10	10	4	1	11	
<input type="checkbox"/>	edit	36	La salade d'oranges de nomra (dattes, ricotta et pistaches)	Découvrez le bon goût de la salade d'oranges à l'ancienne qui vous séduira par ses influences orient...	5	4	4	4	1	13	
<input type="checkbox"/>	edit	37	Tartine tropicale aux fruits et noix de coco	NULL	10	5	5	6	7	9	
<input type="checkbox"/>	edit	38	Scampis à l'ail	NULL	30	20	10	2	7	10	
<input type="checkbox"/>	edit	39	Filets de poulet, mozzarella, épinards et tomates cerises	NULL	72	55	17	6	7	11	
<input type="checkbox"/>	edit	40	Cake au miel	NULL	5	0	5	4	7	12	
<input type="checkbox"/>	edit	41	Smoothie aux myrtilles à l'orange	NULL	15	5	10	4	7	13	
<input type="checkbox"/>	edit	42	Boisson chaude au gingembre et au curcuma	Recette au top	15	10	5	4	4	13	
<input type="checkbox"/>	edit	88	Ma super recette	Healthy Une recette qui vous fera découvrir de nouveau horizon gustatif et de merveilleuses saveurs...	20	10	10	4	4	10	
<input type="checkbox"/>	edit	89	Ma Super Recette Healthy	MaSuperRecette	30	20	10	4	8	12	
<input type="checkbox"/>	edit	90	TestRecetteBeta								

Pour cette partie les difficultés rencontrées ont été de savoir utiliser correctement les relations entre les différentes entités avec notamment le système de l’annotation « **cascad persist** » et principalement le développement et la compréhension de l’algorithme concernant la méthode create du RecipeController de l’API .

Pour résoudre les difficultés auxquelles nous avons été confrontés, nous avons utilisé diverses méthodes, notamment la réalisation de nombreux tests, des recherches sur Internet, la soumission d’issues sur le slack commun de l’école et l'aide précieuse de notre tutrice, Adélie Castel.

## IX. Veille sur les vulnérabilité de sécurité

Durant le projet GoodFood, j'ai effectué une veille spécifique sur la sécurisation des routes concernant la gestion des Access Control Lists (ACL). Technique permettant de garantir la protection de l'application et la confidentialité des données du backoffice et de l'API.

Ainsi en première intention je me suis naturellement dirigé vers la documentation Symfony sur le site <https://symfony.com/>. Ce site permet une recherche par mot clé J'ai ainsi taper « acl configuration » dans la barre de recherche mais les résultats n'étaient pas satisfaisants. J'ai donc saisis simplement le nom du fichier me permettant de coder les paramètres ACL soit « security.yaml ». Je suis entré sur la page <https://symfony.com/doc/current/security.html#form-login> qui m'a permis d'accéder en version 5.4 de Symfony à l'information que je recherchais permettant de comprendre en l'occurrence le fonctionnement des roles hiérarchiques et comment les inscrire sur le fichier de configuration security.yaml.

J'apprends alors que les roles attribués aux utilisateur en BDD peuvent être définis par des règles d'héritage en créant des hiérarchies de role. Un exemple de code est inscrit sur la page en question me permettant d'illustrer et de mieux comprendre ce propos.

Extrait de la page <https://symfony.com/doc/current/security.html#form-login>

**Hierarchical Roles**

Instead of giving many roles to each user, you can define role inheritance rules by creating a role hierarchy:

YAML XML PHP

```
1 # config/packages/security.yaml
2 security:
3     # ...
4
5     role_hierarchy:
6         ROLE_ADMIN:       ROLE_USER
7         ROLE_SUPER_ADMIN: [ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
```

Copy

Users with the `ROLE_ADMIN` role will also have the `ROLE_USER` role. Users with `ROLE_SUPER_ADMIN`, will automatically have `ROLE_ADMIN`, `ROLE_ALLOWED_TO_SWITCH` and `ROLE_USER` (inherited from `ROLE_ADMIN`).

Plus bas se trouve l'explication permettant de définir le chemin d'accès qui sera accessible ou non à un utilisateur en fonction de son rôle, en utilisant les règles de regex pour ce qui concerne le nom de la route.

The screenshot shows a portion of the Symfony documentation for "Securing URL patterns (access\_control)". It includes a code editor with tabs for YAML, XML, and PHP, and a code snippet for security.yaml:

```

1 # config/packages/security.yaml
2 security:
3     # ...
4
5     firewalls:
6         # ...
7         main:
8             # ...
9
10        access_control:
11            # require ROLE_ADMIN for /admin*
12            - { path: '^/admin', roles: ROLE_ADMIN }
13
14            # or require ROLE_ADMIN or IS_AUTHENTICATED_FULLY for /admin*
15            - { path: '^/admin', roles: [IS_AUTHENTICATED_FULLY, ROLE_ADMIN] }
16
17            # the 'path' value can be any valid regular expression
18            # (this one will match URLs like /api/post/7298 and /api/comment/528491)
19            - { path: '^/api/(post|comment)/*', roles: ROLE_USER }

```

De plus sur le lien [https://symfony.com/doc/5.4/security/access\\_control.html](https://symfony.com/doc/5.4/security/access_control.html), se trouve d'autres informations apportant des informations concernant des options de correspondances supplémentaires comme la méthode **path** que nous avons utilisé.

### 1. Options de correspondance

Symfony crée une instance de `RequestMatcher` pour chaque `access_control` entrée, qui détermine si un contrôle d'accès donné doit être utilisé ou non sur cette requête. Les `access_control` options suivantes sont utilisées pour la correspondance :

- `path`: une expression régulière (sans délimiteurs)
- `ip` ou `ips` : les masques de réseau sont également pris en charge (peut être une chaîne séparée par des virgules)
- `port`: un nombre entier
- `host`: une expression régulière
- `methods`: une ou plusieurs méthodes

Voir image ci-dessous :

Après avoir lu et compris les informations recueillies lors de cette veille sur la sécurité, j'ai décidé d'expliquer ces concepts à mes collègues et de mettre en place la configuration dans le fichier `security.yaml`. Pour conclure cette veille m'a permis d'acquérir une bonne compréhension du concept d'ACL dans le contexte de Symfony et de mettre en œuvre les mesures appropriées pour sécuriser les routes de notre projet sur le backoffice et l'API.

## X. Problèmes rencontrés et résolutions

Lors de l'élaboration du projet nous avons rencontré quelques problématiques qui nous ont pris du temps à résoudre.

### 1. Easy Admin

Lors de la phase de conception du backoffice nous avions décidé de mettre en place le composant Easy Admin permettant de créer un backoffice design et disposant de nombreuses fonctionnalités préétablies. Malheureusement après de longues sessions de débogage nous avons décidé d'arrêter le développement de ce composant. Steven avait développer sur une autre branche un CRUD via la commande Symfony **bin/console make:crud**. Ainsi nous avons conçu un backoffice manuellement mais tout autant fonctionnelle pour l'utilisation prévu sur le MVP.

### 2. Le Champ ‘Steps’

Le champ Steps nous a posé problème au départ car nous l'avions intégré dans la table ‘Recipe’ sous format : **array**. Le type ‘array’ nous permettait théoriquement de rentrer plusieurs étapes d'une recette afin de les réemployer plus facilement dans l'application. Or pour la manipulation des données et la sérialisation le type ‘array’ n'a jamais pu être correctement exploité et a occasionné de nombreux bug sur l'application côté backend. Pour solutionner ce problème nous avons décidé en toute logique de créer une table/entité à part que nous avons appelé Steps et qui comprends un champ ‘content’ définissant à chaque fois une des étapes d'une recette.

### 3. L'upload d'image

De nombreux bugs ont été occasionné en voulant développer la fonctionnalité de téléchargement d'image. L'un des plus importants a été de saisir que le fichier que renvoyait le frontOffice via le formulaire d'ajout de recette n'était pas au format json mais de type file. Il a donc fallu créer deux routes distinctes côté front une pour transmettre les informations de la recette en format Json et l'autre pour transmettre l'image en format de type file. Pour l'algorithme sur le controller de l'API côté backend c'est grâce à notre référente Adélie Castel que l'on a pu trouver une solution à ce problème.

**Fonctionnalités manquantes au projet pour la finalisation du MVP :**

- Formulaire de Modification d'une recette pour un utilisateur côté FrontOffice
- Fonctionnalité d'Autosuppression d'une image côté Backend

## XI. Extrait d'une documentation d'un site en anglais

Texte extrait du PDF **owasp\_MASVS** concernant le document de présentation : OWASP Mobile Application Security Verification Standard v2.0.0

### Original version

#### About the Standard

The OWASP Mobile Application Security Verification Standard (MASVS) is the industry standard for mobile application security. It provides a comprehensive set of security controls that can be used to assess the security of mobile apps across various platforms (e.g., Android, iOS) and deployment scenarios (e.g., consumer, enterprise). The standard covers the key components of the mobile app attack surface including storage, cryptography, authentication and authorization, network communication, interaction with the mobile platform, code quality and resilience against reverse engineering and tampering. The OWASP MASVS is the result of years of community effort and industry feedback. We thank all the contributors who have helped shape this standard. We welcome your feedback on the OWASP MASVS at any time, especially as you apply it to your own organization and mobile app development projects. Getting inputs from a variety of mobile app developers will help us improve and update the standard which is revised periodically based on your inputs and feedback. You can provide feedback using GitHub Discussions in the OWASP MASVS repo <https://github.com/OWASP/owasp-masvs/discussions>, or contact the project leads directly <https://mas.owasp.org/contact/>. The OWASP MASVS and MASTG are trusted by the following platform providers and standardization, governmental and educational institutions.

### Traduction Française

#### A propos

La norme de vérification de la sécurité des applications mobiles OWASP (MASVS) est la norme de l'industrie pour la sécurité des applications mobiles. Il fournit un ensemble complet de contrôles de sécurité qui peuvent être utilisés pour évaluer la sécurité des applications mobiles sur diverses plates-formes (par exemple, Android, iOS) et des scénarios de déploiement (par exemple, grand public, entreprise). La norme couvre les composants clés lors d'attaque sur des applications mobiles, notamment le stockage, la cryptographie, l'authentification et l'autorisation,



la communication réseau, l'interaction avec la plate-forme mobile, la qualité du code et la résilience contre l'ingénierie inverse et la falsification. L'OWASP MASVS est le résultat d'années d'efforts communautaires et de commentaires de l'industrie. Nous remercions tous les contributeurs qui ont contribué à façonner cette norme. Nous apprécions vos commentaires sur le OWASP MASVS à tout moment, en particulier lorsque vous l'appliquez à votre propre organisation et à vos projets de développement d'applications mobiles. Obtenir des contributions de divers développeurs d'applications mobiles nous aidera à améliorer et à mettre à jour la norme qui est révisée périodiquement en fonction de vos contributions et de vos commentaires. Vous pouvez fournir des commentaires à l'aide des discussions GitHub dans le référentiel OWASP MASVS <https://github.com/OWASP/owasp-masvs/discussions>, ou contacter directement les responsables du projet <https://mas.owasp.org/contact/>. L'OWASP MASVS et le MASTG sont approuvés par les fournisseurs de plateformes et les institutions de normalisation, gouvernementales et éducatives suivantes.

## XII. Conclusion

En conclusion, je dirais que l'expérience de l'Apothéose, nom excellamment bien choisi par l'école O'Clock pour notre projet de fin de formation, a été une grande expérience d'apprentissage tant sur le plan technique que sur le plan de l'humain.

- Au niveau technique, j'ai pu améliorer mes connaissances en découvrant de nouveaux concepts et outils tels que Ant-Design pour React, Insomnia pour tester les requêtes ou encore Easy Admin que nous avons envisagé pour la création du backoffice.
- J'ai également pu constater que les compétences acquises au cours de la formation ont pu être pratiquées malgré les nombreuses difficultés rencontrées, ce qui m'a donné plus confiance en moi.
- J'ai ressenti un grand plaisir d'accomplissement et de satisfaction en constatant les résultats de mon travail. Notamment après la résolution de bug, la création d'algorithme ou plus globalement la compréhension du code et de sa logique.
- En outre, ce projet m'a donné l'opportunité de travailler selon la méthode Agile Scrum. En travaillant en tant que product owner, j'ai pu mieux comprendre les rouages de cette méthode et sa mise en œuvre pratique, telle que la planification des sprints, la tenue des réunions quotidiennes et la mise en place d'un backlog produit.
- Sur le plan humain, j'ai beaucoup apprécié le travail d'équipe, la collaboration et la création de liens avec mon équipe. L'ambiance était très agréable et nous avons réussi à trouver un bon équilibre entre nos compétences respectives et nos soft skills, ce qui pour moi a beaucoup contribuer à la réussite du projet GoodFood.

De plus, étant donné l'intérêt suscité par GoodFood auprès de mon entourage en raison de son design, de son utilisation facile et de son concept novateur, je prévois de poursuivre ce projet en dehors de la formation avec l'équipe initiale, dans la mesure du possible. L'objectif est de développer la version 2 du projet telle qu'elle est définie dans le cahier des charges et de déployer finalement l'application.

Enfin, je souhaitais prendre un moment pour remercier l'école O'Clock pour la formation incroyable qui m'a été apportée et mes tuteurs Ludovic Argenty et Adélie Castel pour leur aide précieuse tout au long du projet ; mon équipe de développement Alexis, Abdoul, Steven et Gael, qui ont fait un excellent travail pour atteindre nos objectifs et finaliser la V1 de GoodFood. Je tiens enfin à remercier mon entourage pour leur soutien tout au long de cette aventure. Sans leur encouragement et leur présence à mes côtés, je n'aurais pas pu relever les défis de ce nouveau cursus et présenter avec fierté ce projet aujourd'hui. Merci à eux pour leur soutien inconditionnel !

# Annexes

# Rendu Visuel de l'application GoodFood

## 1. Page d'accueil du FrontOffice: Version desktop



## 2. Espace Mon Profil : Version mobile



### 3. Page des entrées : Version tablette

GoodFood

Rechercher un aliment

Déconnexion

Accueil

#### Entrée

Image	Nom de l'entrée
	Soupe de millet au saumon, poireaux et oeuf dur
	Sardines à la plancha, huile d'olive à la marjolaine
	Oeufs brouillés crémeux
	Morning Champion
	Champignons farcis
	Salade performance
	Soupe de légume vert
	Gaspacho

# Wireframes élaborés par l'équipe:

Home Page



Nav Bar

logo

Search bar

Ajout recette

Description site

Recette aléatoire

Titre de la recette

Titre de la recette

Vertues

Vertue

Vertue

Vertue 1

Vertue

Contactez nous

Mentions légales

Admin

Recette

Catégorie 1 Catégorie 2 Catégorie 3 Catégorie 4 Catégorie 5

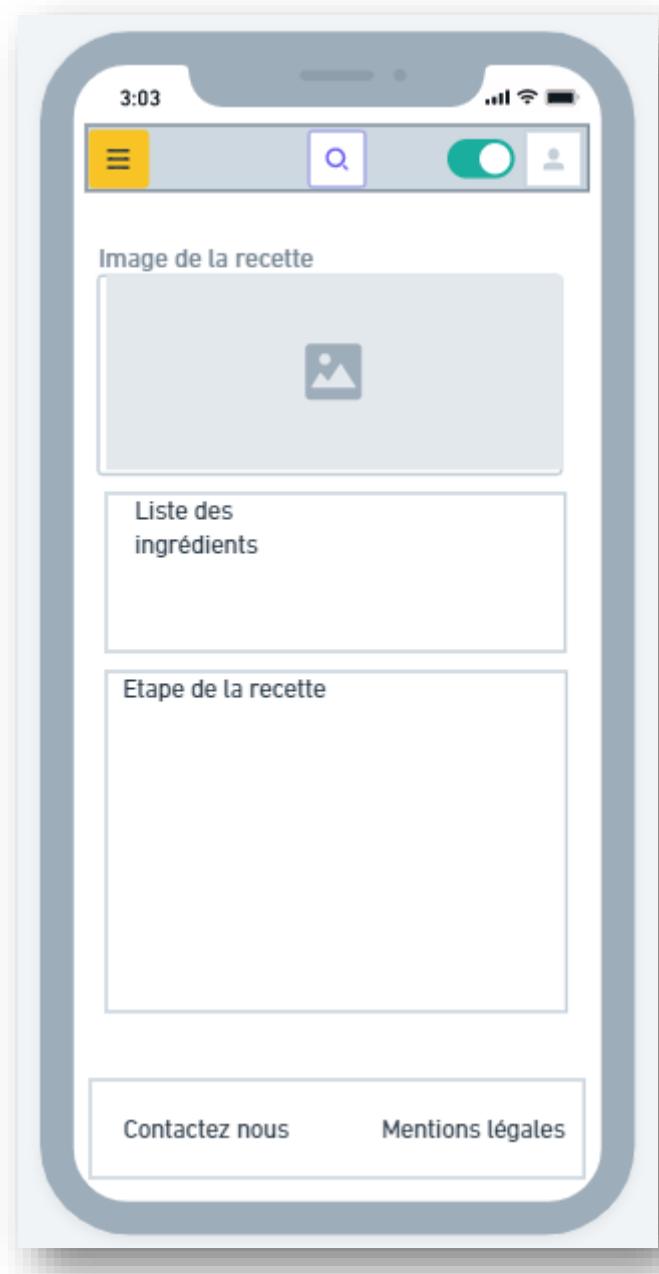
Vetues

Catégorie 1 Catégorie 2 Catégorie 3 Catégorie 4  
Catégorie 5 Catégorie 6 Catégorie 7 Catégorie 8

Liste des recettes



## Recette



Nav Bar

☰

Search icon

User icon

Liste des ingrédients

Image de la recette

Etape de la recette

Contactez nous

Mentions légales

## Liste des recettes



Nav Bar

image

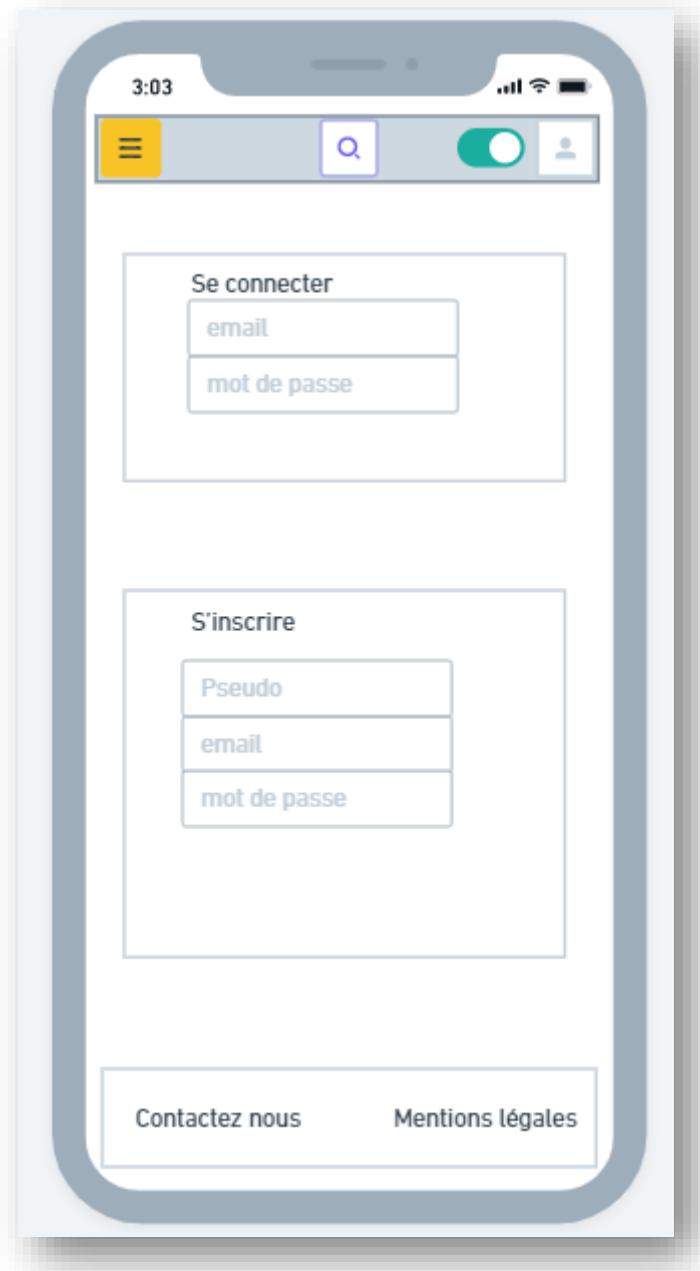
titre recette

1      2

Contactez nous      Mentions légales



## Connexion



Nav Bar

≡

Q

Toggle

User icon

Se connecter

email

mot de passe

S'inscrire

pseudo

email

mots de passe

Contactez nous

Mentions légales



## Page ajout recette



Nav Bar

≡

Search icon

Toggle switch

User profile icon

### Ajouter une recette

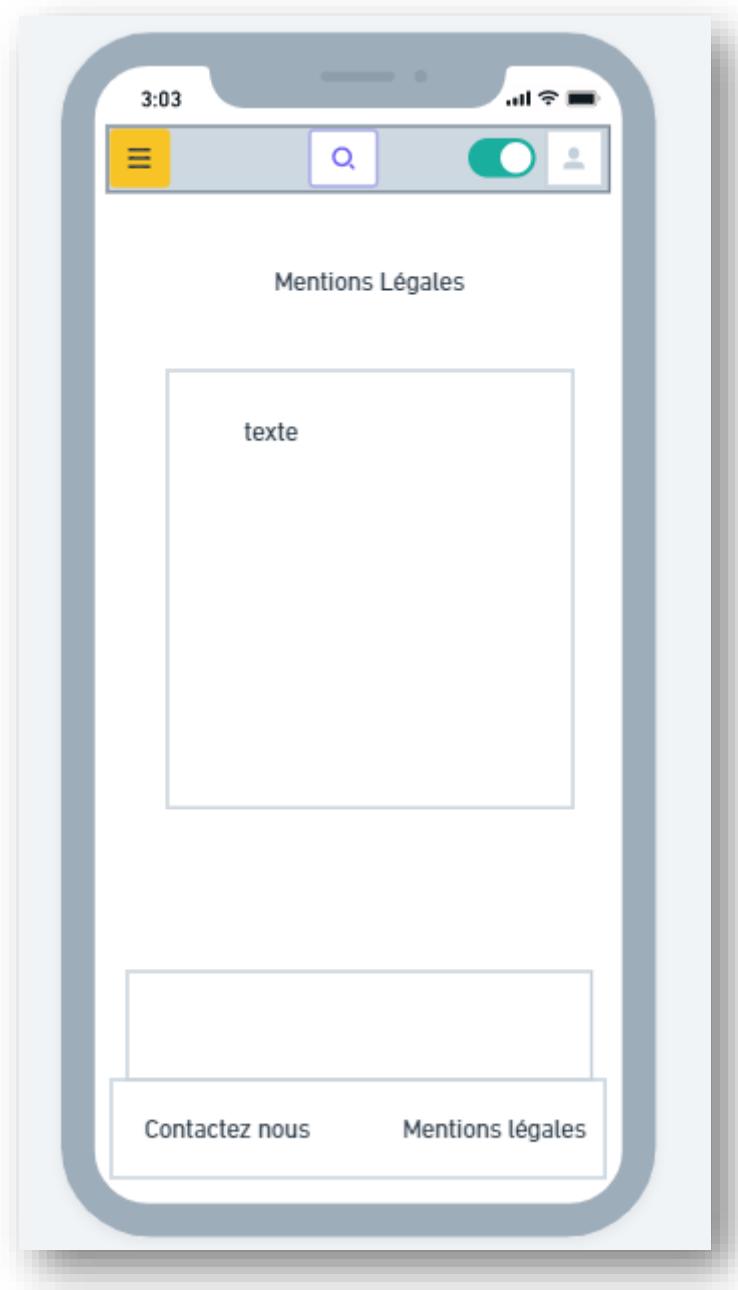
• Titre de votre recette	Votre titre
• Description de la recette	Description
• Ingrédients/Unité/Quantité	Liste des Ingredients/Unité/Quantité
• Vertue de votre recette	Vertue A selectionner
• Durée de préparation	Durée
• Nombre de convive	Nb de personne
• Première étape de la recette	1iere Etape
• Deuxième étape de la recette	2e Etape
• Troisième étape de la recette	3eme Etape
• Quatrième étape de la recette	4eme Etape
• Cinquième étape de la recette	5eme Etape
• Photo de votre recette	Ajout de fichier image

Contactez nous

Mentions légales



## Page Mentions légales



Nav Bar

☰

🔍

Toggle

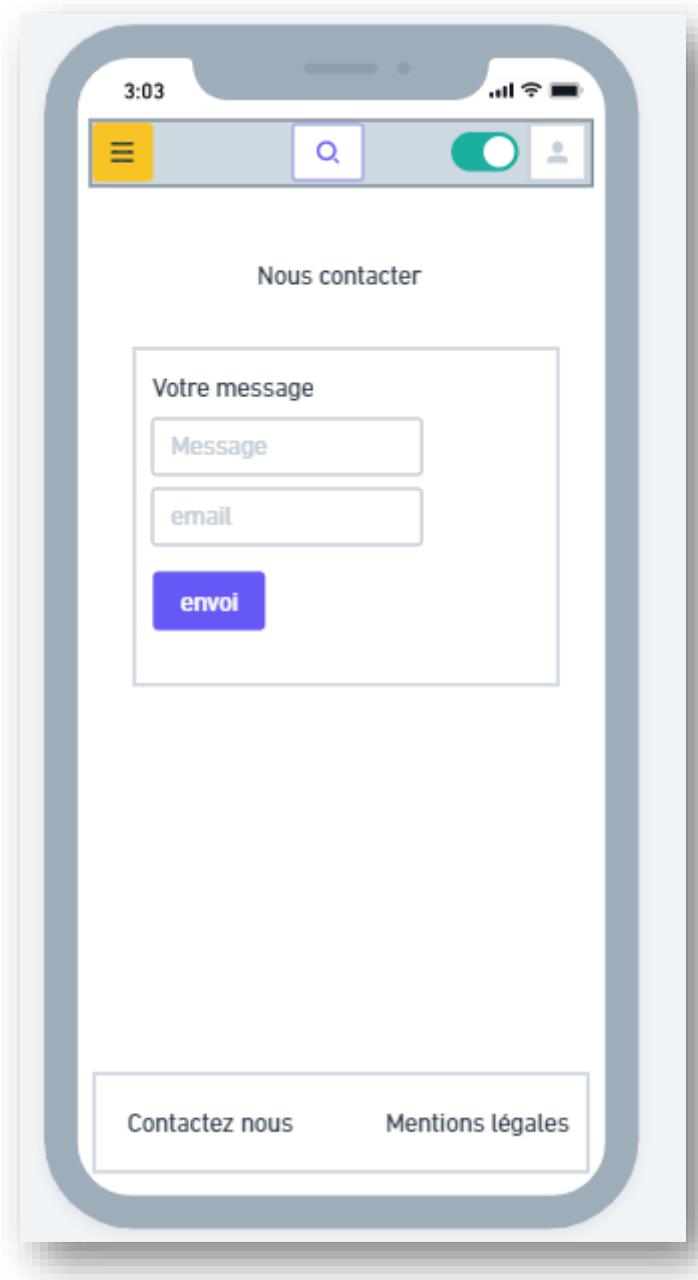
👤

Mentions Légales

texte

Contactez nous      Mentions légales

## Page Contactez Nous



Nav Bar

☰

Q

Toggle

User icon

Nous contacter

Votre message

Message

email

envoi

Contactez nous      Mentions légales



## Page Catégorie



Nav Bar

☰

SEARCH

ON/OFF

USER PROFILE

CATEGORIE

image

titre recette

image

titre recette

image

titre recette

image

titre recette

1

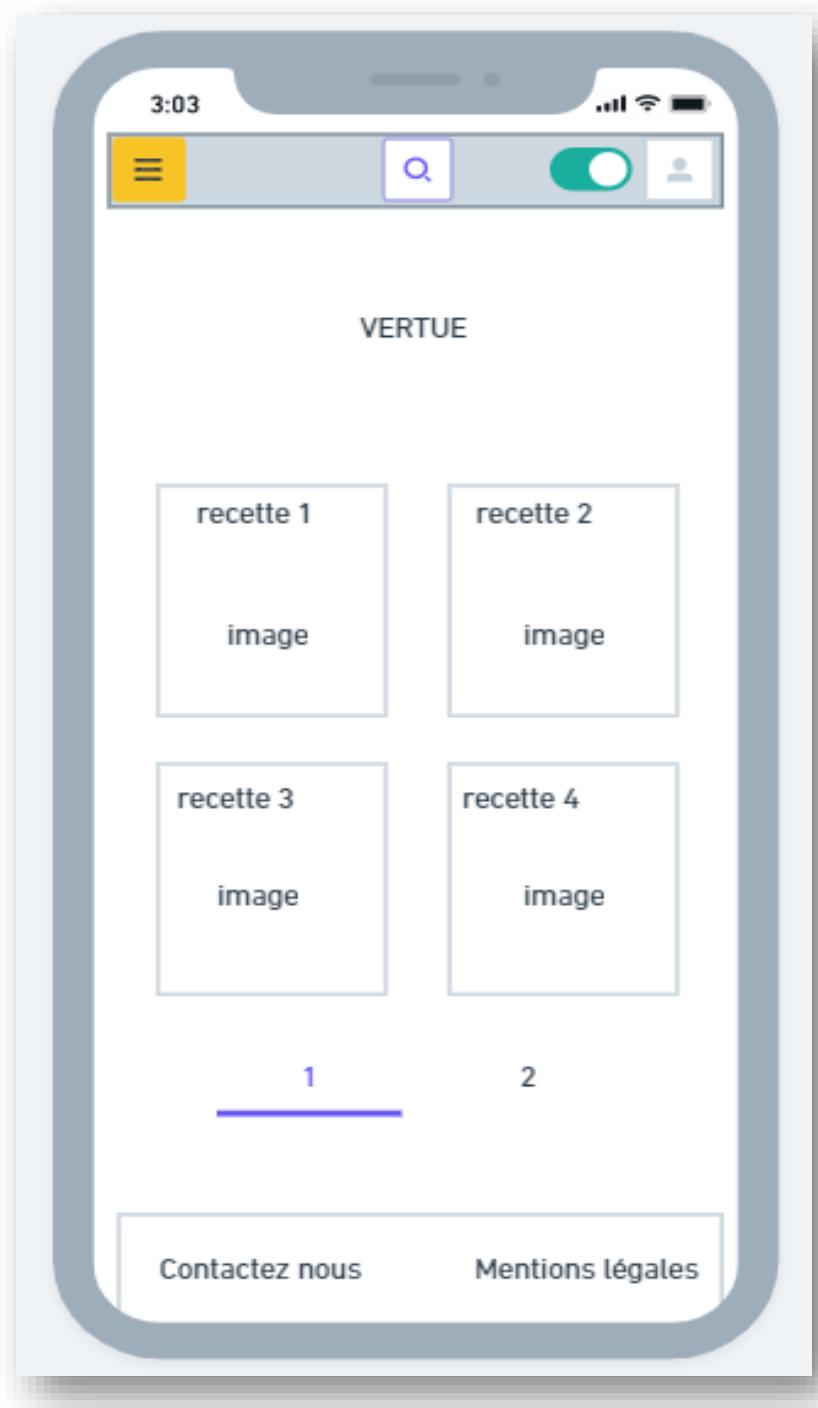
2

Contactez nous

Mentions légales



## Page Vertu



Nav Bar

VERTUE

image

titre recette

image

titre recette

image

titre recette

image

titre recette

1

2

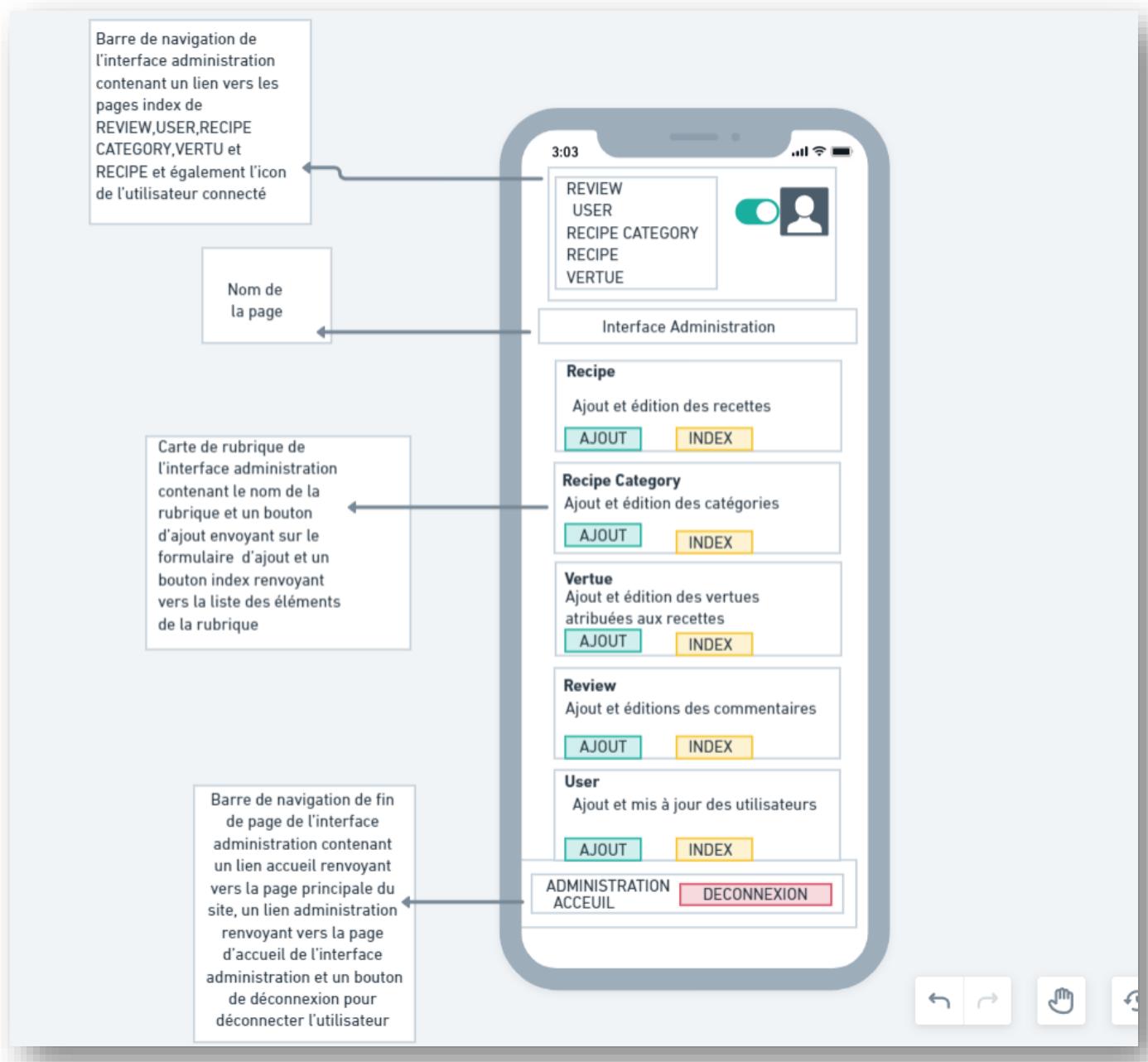
Contactez nous

Mentions légales



## Wireframe Back

### Home Page



The screenshot shows a web-based administration interface with the following layout:

- Header:** A navigation bar with tabs: RECIPE, RECEPTE CATEGORY, VERTUE, REVIEW, and USER. To the right of the tabs are a toggle switch and a user icon.
- Section Header:** A box labeled "Interface Administration".
- Content Area:** Five boxes representing different administrative modules:
  - Recipe:** Adds and edits recipes. Buttons: AJOUT (green), INDEX (yellow).
  - Recipe Category:** Adds and edits categories. Buttons: AJOUT (green), INDEX (yellow).
  - Vertue:** Adds and edits virtues assigned to recipes. Buttons: AJOUT (green), INDEX (yellow).
  - Review:** Adds and edits reviews. Buttons: AJOUT (green), INDEX (yellow).
  - User:** Adds and updates users. Buttons: AJOUT (green), INDEX (yellow).
- Footer:** A navigation bar with links: ACCEUIL, ADMINISTRATION, and DECONNEXION (highlighted in red).



## Page d'ajout

RECIPE    RECIPE CATEGORY    VERTUE    REVIEW    USER   

Formulaire d'ajout d'une nouvelle recette

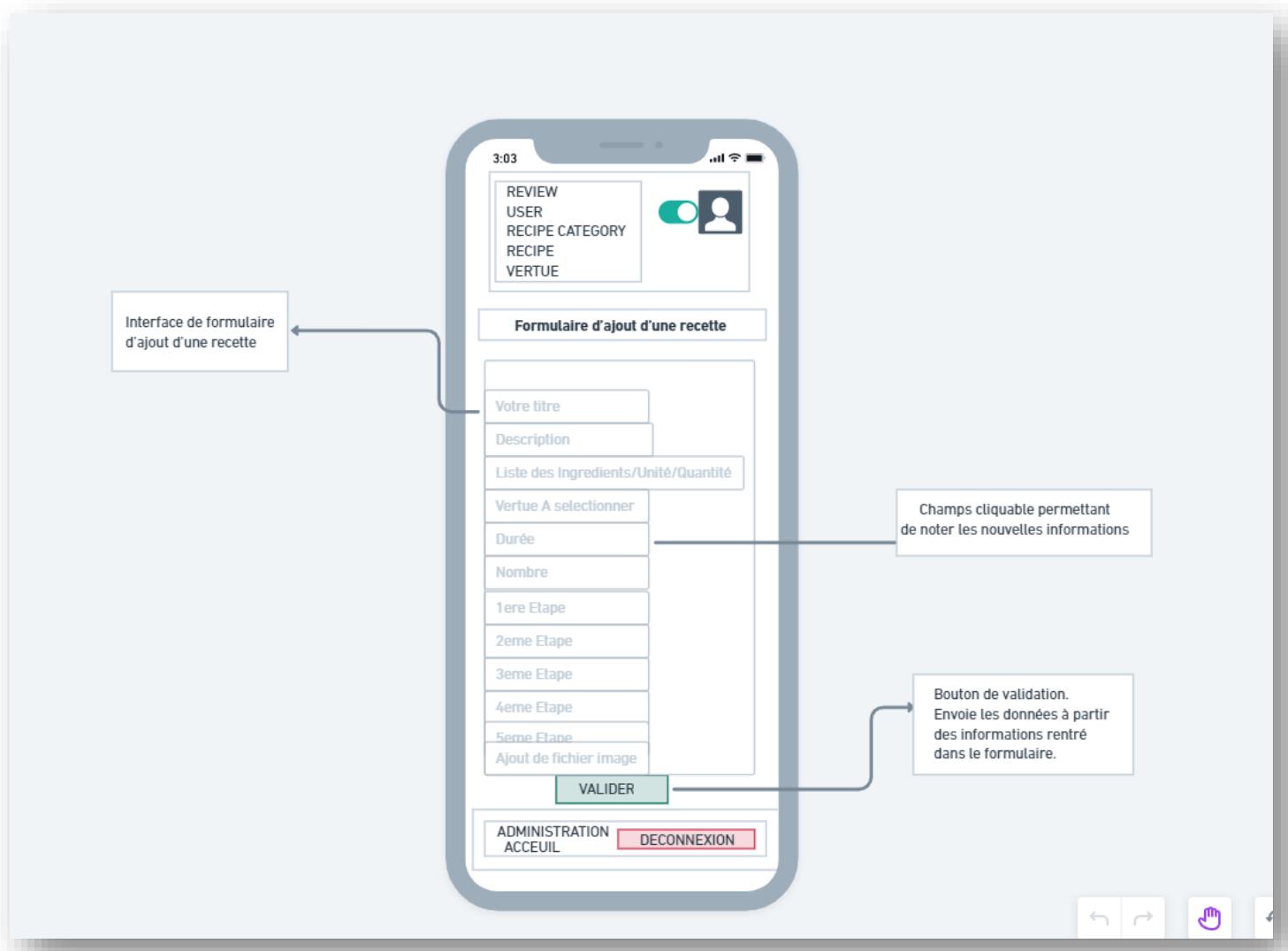
Le design de la page sera préservé pour tous les formulaires de la partie administration

• Titre de votre recette	Votre titre
• Description de la recette	Description
• Ingrédients/Unité/Quantité	Liste des Ingredients/Unité/Quantité
• Virtue de votre recette	Vertue A selectionner
• Durée de préparation	Durée
• Nombre de convive	Nombre
• Première étape de la recette	1ere Etape
• Deuxième étape de la recette	2eme Etape
• Troisième étape de la recette	3eme Etape
• Quatrième étape de la recette	4eme Etape
• Cinquième étape de la recette	5eme Etape
• Photo de votre recette	Ajout de fichier image

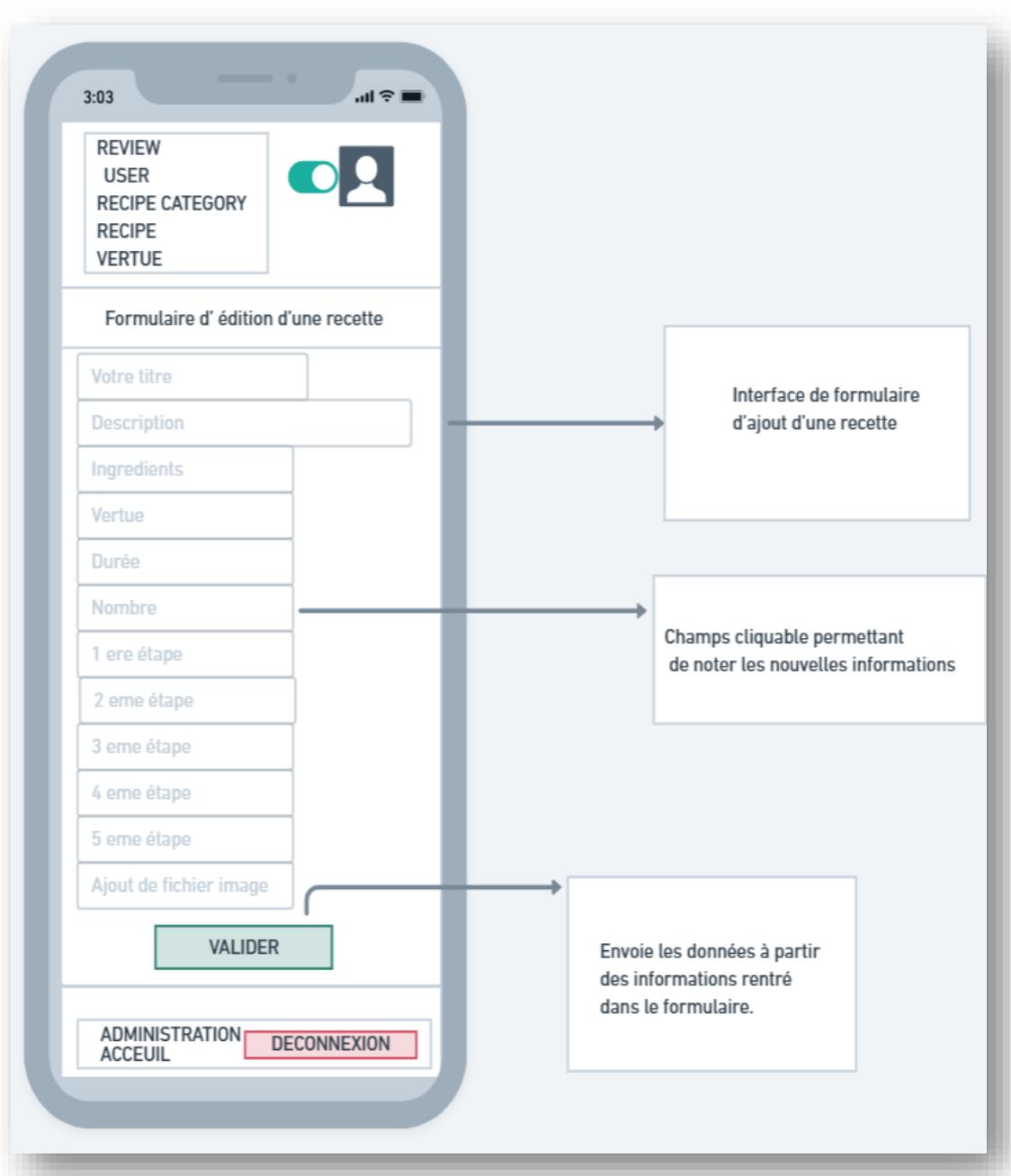
VALIDER

ACCEUIL    ADMINISTRATION    DECONNEXION





## Page d'édition



RECIPE
RECIPE CATEGORY
VERTUE
REVIEW
USER



Formulaire Édition d'une recette

- Titre de votre recette
- Description de la recette
- Ingrédients/Unité/Quantité
- Vertue
- Durée de préparation
- Nombre de convive
- Première étape de la recette
- Deuxième étape de la recette
- Troisième étape de la recette
- Quatrième étape de la recette
- Cinquième étape de la recette
- Photo de votre recette

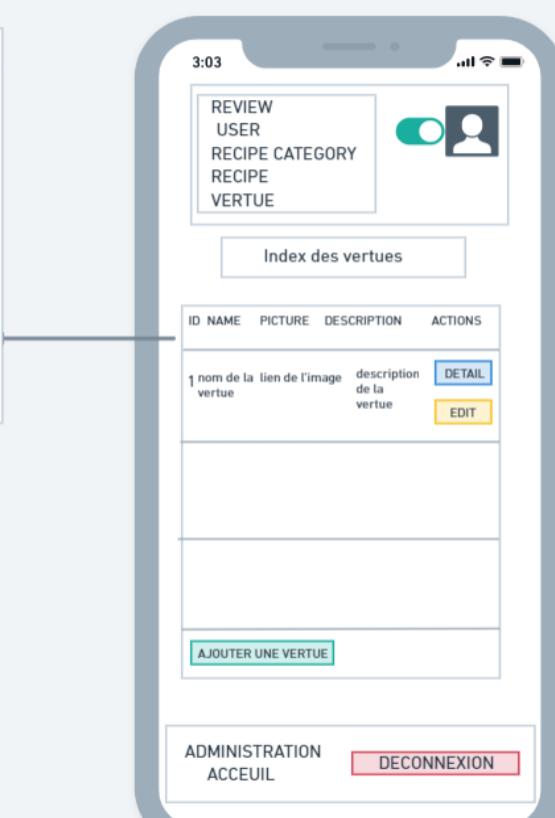
VALIDER

ACCEUIL
ADMINISTRATION
DECONNEXION

## Page d'index des vertues

Tableaux contenant la liste des vertues et affichant leur ID, leur nom, le lien de l'image, la description de la vertue , ainsi que un bouton détaill permettant de visualiser le DETAIL d'une vertue et un bouton EDIT renvoyant vers le formulaire de modification et un bouton AJOUTER renvoyant vers le formulaire d'ajout .

Cette page sera la même pour toutes les rubriques de l'interface administration seul les données affichées seront momifier



RECIPE   RECIPE CATEGORY   VERTUE   REVIEW   USER



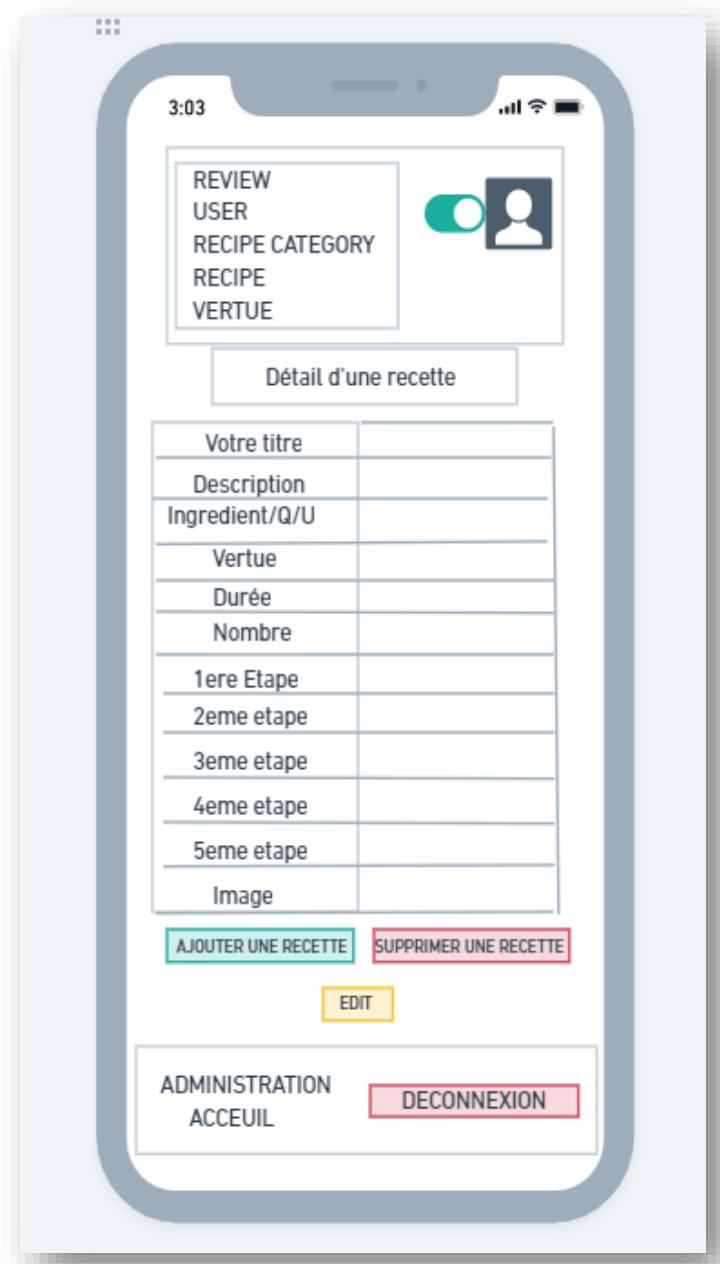
Index des vertues

ID	NAME	PICTURE	DESCRIPTION	ACTION
1	nom de la vertue	lien de l'image	description de la vertue	<button>DETAIL</button> <button>EDIT</button>
<a href="#">AJOUTER UNE VERTUE</a>				

ACCEUIL   ADMINISTRATION   DECONNEXION



## Page détails d'une recette



RECIPE    RECIPE CATEGORY    VERTUE    REVIEW    USER     

Détail d'une recette

Votre titre	
Description	
Ingredient/Q/U	
Vertue	
Durée	
Nombre	
1ere Etape	
2eme etape	
3eme etape	
4eme etape	
5eme etape	
Image	

[SUPPRIMER UNE RECETTE](#)

[AJOUTER UNE RECETTE](#)

[EDIT](#)

ACCEUIL    ADMINISTRATION    [DECONNEXION](#)

