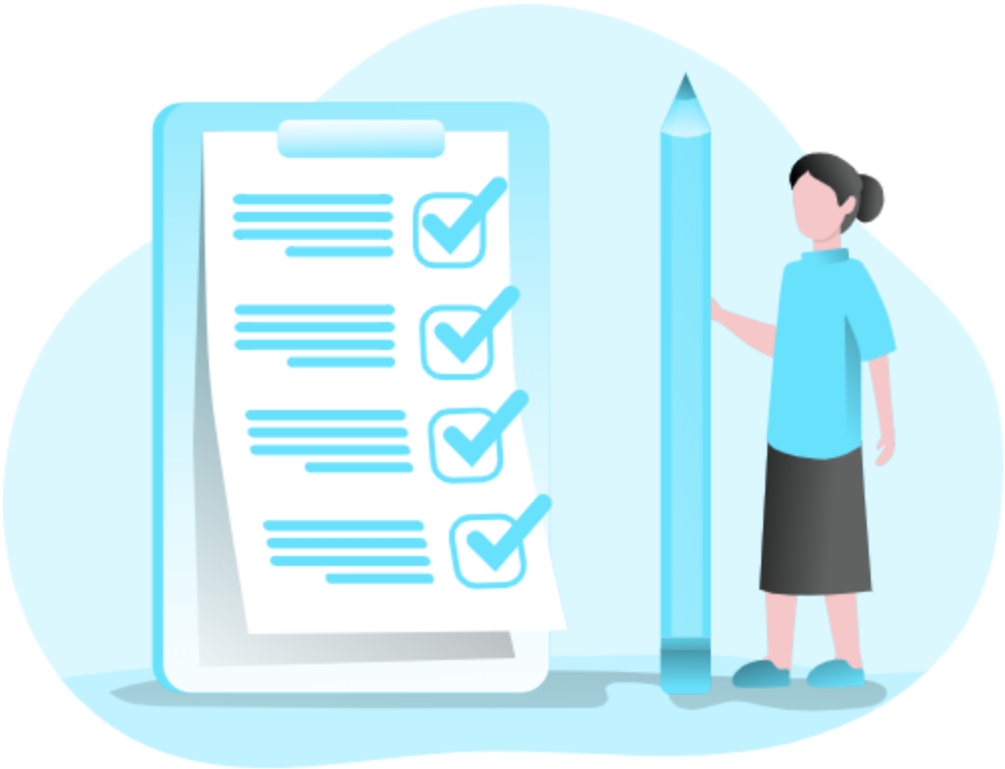
Dossier Technique

Food Sens

Quentin Razavet, Louison Boutet et Loïc Bigot

11 Novembre 2022

Sommaire



Introduction du Projet

Food Sens est une application web visant à aider ses utilisateurs au quotidien dans l'élaboration de leur régime alimentaire. En leur proposant des recettes riches et variées adaptées aux ingrédients dont ils disposent en temps réel.

Nous voulons mettre la cuisine à la portée de tous et vous donner les outils afin de mieux la comprendre et suivre vos apports nutritionnels quotidiens.

Contexte

Nous sommes tous les trois en troisième année de Licence Informatique à la Faculté des Sciences et Techniques de Tours . Nous avons mené ce projet dans le cadre de notre cours relatif au Développement Web .

Nous avons profité de l'opportunité d'avoir une liberté totale quant au choix du sujet, pour réaliser une idée de projet personnel que nous avions déjà eu dans le passé.

Clients Cibles

La clientèle pour notre projet est vaste et ciblée à la fois, nous simplifions et banalisons l'accès à la cuisine avant tout pour la mettre à portée de tous.

Toutefois nous pouvons reconnaître des profils types pour qui notre application apporte une valeur ajoutée :

- **les étudiants**
- **les sportifs**

Ces profils sont susceptibles de trouver un fort intérêt pour notre projet. Car nous permettons de faciliter l'accès à un régime alimentaire simple et varié pour les plus débutants, tout en mettant à disposition des outils de calculs d'apports nutritionnels pour les plus minutieux.

Cahier des charges

A travers ce projet nous de nombreux objectifs quant aux principales fonctionnalités à implémenter :

- **Interface utilisateur + Interface d'administrateur**
- **Communication optimale avec notre BDD**
- **Interfaces Accueil, Votre Frigo, Recherche de Recette, Vos Favoris, Recap Journalier**
- **Algorithmes relatifs au différentes interfaces :**
    - **recherche de recette**
    - **ajouter / retirer des favoris**
    - **créer des ingrédients/recettes/type d'ingrédient**
- **Uploads d'images (pour les images de recettes + image de profil utilisateur + image d'ingrédients)**
- **Envoie de Mail**
- **Modification de profil + Suppression du compte**
- **Connexion anonyme + Bloquer une connexion d'utilisateur**
- **Commentaires de recette + modification et suppression**
- **Système de liste de course**
- **Formulaire de contact**
- **Une accessibilité UX / UI pour le design graphique**

Environnement de développement

-GitHub(github.com/docokara/FoodSens) et GitHub Desktop, afin de permettre à chacun de travailler indépendamment sur les branches auxquelles il est attribué sans perturber le travail des autres tout en faisant des merges à chacune des avancées majeures.

-VisualStudio Code avec différentes extensions tel que LiveShare pour coder ensemble en même temps , Prettier pour l'indentation uniformisée du code.

-Wamp pour la mise en place d'un Server APACHE , MYSQL et d'open ssl inclus dedans

-Symfony 6.1
-PHP 8.1.6
-PhpMyAdmin 5.1.1
-Discord pour le partage d'informations et l'organisation du projet

Description des différentes technologies utilisées

Durant la conception de Food Sens, nous avons utilisés différentes fonction/librairies :

-Fonction Mailer

-Prettier

-UserPasswordHasher

-File Uploader

Présentation du framework Symfony

Symfony est une collection de composants PHP et un framework MVC gratuit écrit en PHP. Il offre des fonctionnalités modulaires et personnalisables qui facilitent et accélèrent le développement de sites Web.

Symfony fut créé en 2005 pour faciliter les développeurs web. SensioLabs, à l'origine de Symfony, voulait créer un nouveau framework pour ne plus avoir à refaire les mêmes tâches en boucle telles que l'implémentation de gestions d'utilisateurs, le mapping objet-relationnel(ORM) ou autre. Comme beaucoup de développeurs avaient les mêmes besoins, le framework a été partagé à la communauté PHP.

Écrit en PHP, Symfony marche sur les 3 systèmes d'exploitations majeurs (MacOS, Windows et Linux) et est disponible dans plusieurs langues. Il est maintenu depuis plus de 15 ans et est toujours mis à jour (la dernière mise à jour date du 12 octobre 2022). Il a eu pour le moment, 6 versions majeures qui évoluent avec les nouvelles versions de PHP.

Pour améliorer la maintenabilité et l'évolutivité, le code est séparé en trois couches selon le modèle MVC avec des performances optimisées et un système de mise en cache pour garantir des temps de réponse optimaux.

Une gestion des URL est aussi mise à disposition. Elle permet aux pages d'avoir des URL différentes de leur position dans l'arborescence, un système de configuration en cascade qui utilise entièrement le langage de description YAML. Ceci permet aussi une meilleure gestion du back-end.

Symfony a une prise en charge d'AJAX pour les différentes requêtes vers les serveurs web (JavaScript et XML) et une architecture extensible qui permet la création et l'utilisation de plugins.

Symfony est très pratique grâce à sa simplicité et toute la documentation très fournie sur son site web avec une diversification très importante dans beaucoup de domaines via les modules et les plugins.

Procédure d’installation

Pour ouvrir et utiliser notre application vous allez devoir procéder à quelques manipulations :

- composer update && composer install

- modifier le .env /mettre le bon URL local vers votre BDD

- créer une BDD local vide intitulée “foodsens”

- symfony console make:migration && symfony console doctrine:migration:migrate

- Pour granted un user au rôle d’admin changer son rôle dans la base de donnée à : [“ROLE_ADMIN”]

Architecture et conception

Notre application dispose d’une architecture simple, afin de faciliter son utilisation. Cela se traduit par un site One Page dans lequel figure l’interface principale divisée en trois main différents dans lesquels nous faisons apparaître les différentes informations en fonction de :
-la page dans laquelle il se trouve

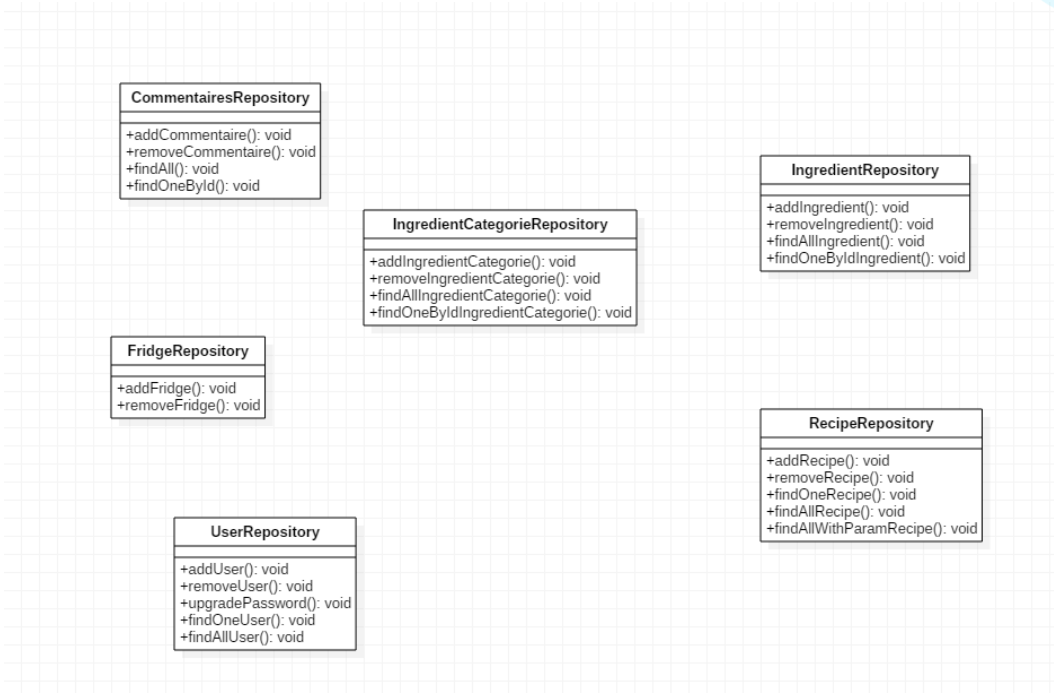
-les informations envoyées par le back au front

-si l’utilisateur est connecté ..

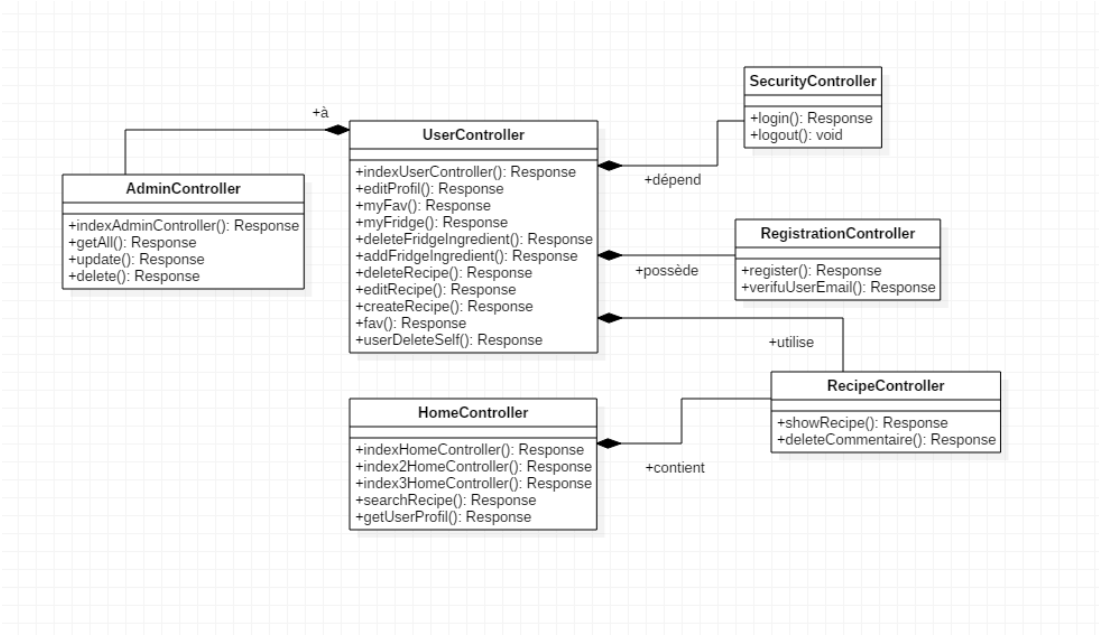
Modèle de données

Nos modèle de données peut être schématisé de la façon suivante :

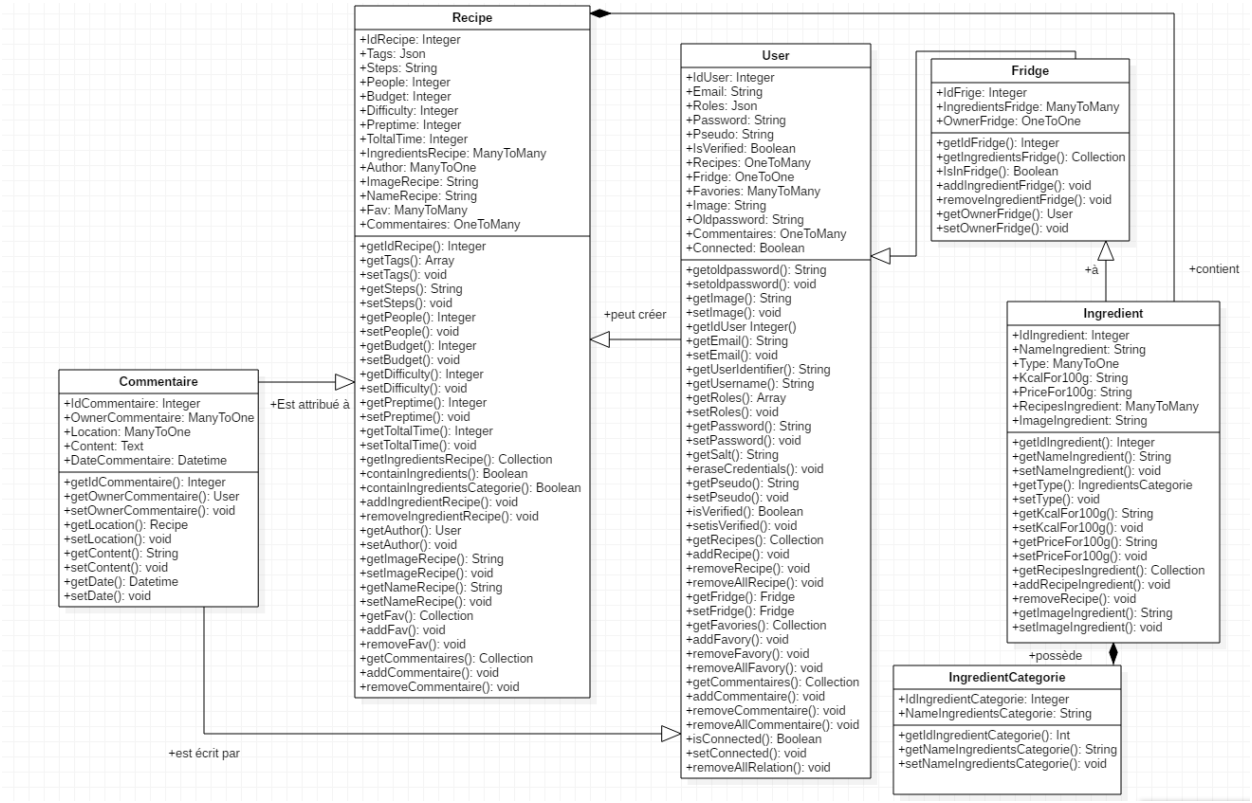
Repository :



Controller :



Entity :



Tandis que la communication des requêtes FrontToBack ressemble à :

Découpages des controllers

Notre application possède 6 controllers :

AdminController :

- **fonction de route d'index**
- **fonction d'affichages des utilisateurs**
- **update / delete users**

HomeController :

- **Affichage des différentes rubriques : notre équipe, nous contacter et notre projet**
- **filtre de recherche des recettes**
- **accès au page personnel d'un user (ses commentaires , ses favoris)**

RecipeController :

- **fonction d'affichage de recette**
- **fonction d'édition de commentaire**

RegistrationController :

- **fonction register**
- **fonction d'envoie de mail de vérification**

SecurityController :

- **route pour s'enregistrer avec envoie de mails de confirmation**
- **route de callback de confirmation d'email**

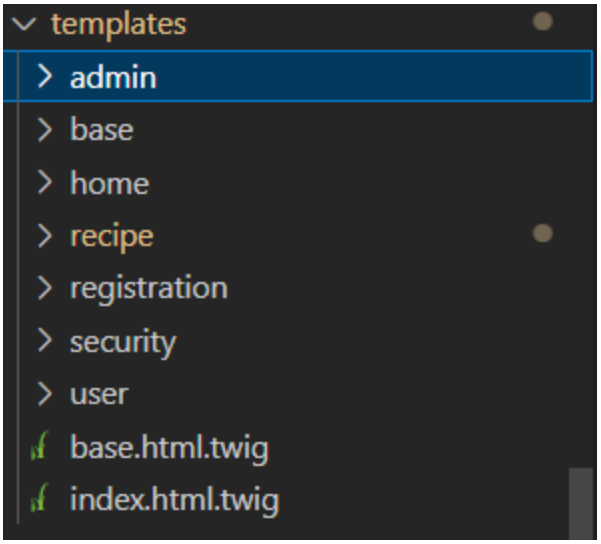
UserController :

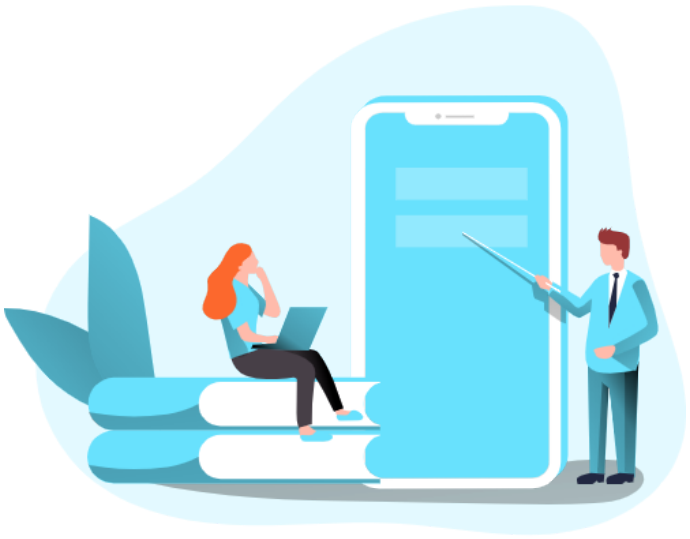
- **fonction d'affichage de la page de profil utilisateur**
- **fonction edit profile**
- **fonction relative aux favoris de l'utilisateur**
- **frigo utilisateur**
- **création/ modification des recettes crées**

Présentation du templating

Nous avons opté pour un site Single Page application. Par conséquent, une majorité de nos templates sont importés dans /templates/index.html.twig qui en fonction de l'état que renvoie le controller affiche les informations adéquates.

Grâce à ce choix, cela nous permet d'avoir toujours la même page (header,footer,aside nav) tout en modifiant son contenu.





Gestion de la sécurité et des rôles utilisateurs

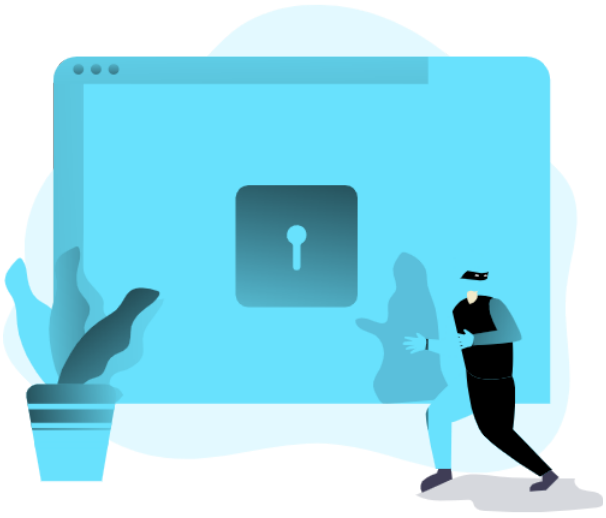
Les utilisateurs peuvent avoir 4 rôles différents, ROLE_ADMIN, ROLE_MODO, ROLE_BLOCKED et ROLE_USER. Par défault un utilisateur connecté possède le rôle : ROLE_USER

Pour plus de sécurité, lors de la création d'un utilisateur, son mot de passe est hashé dans la base de données pour ne pas être récupérable, même par les admins.

Pour couper l'accès à différentes sections du site tel que /admin, le "ROLE_ADMIN" est requis pour y accéder.

Lorsqu'on possède le "ROLE_BLOCKED" le site devient complètement inaccessible pour les personnes lors de leur connexion.

```yaml
    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used
    access_control:
        - { path: ^/admin, roles: ROLE_ADMIN }
        - { path: ^/, allow_if: "not is_granted('ROLE_BLOCKED')" }
when@test:
    security:
        password_hashers:
            # By default, password hashers are resource intensive and take time. This is
            # important to generate secure password hashes. In tests however, secure hashes
            # are not important, waste resources and increase test times. The following
            # reduces the work factor to the lowest possible values.
            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
```



Solution d’upload des fichiers

Nous avons ajouté un service nommé :

App\Service\FileUploader qui possède comme target directory le dossier /public/uploads/ afin d’y stocker les images et d’y avoir accès.

Ensuite nous ajoutons FileUploader.php dans le dossier src/Service

Ensuite il suffit d’importer FileUploader dans un controller et appelé sa fonction upload qui prend comme paramètre un file.

Celui-ci prend un nom unique qui est ensuite attribué comme image à l’utilisateur dans la base de données.

Configuration des librairies communautaire

Durant ce projet nous nous sommes servi de différentes librairies communautaires, telles que:

- **Prettier**
- **UserPaswordEncoder**
- **FileUploader**
- **EasyAdmin**
- **Et bien sûr Symfony dans son ensemble**

Leur configuration dans le projet se traduit par des importants dans les différents controllers dans lesquels elles sont utilisées.

Structure des fichiers de style

Notre application dispose de 3 feuilles de style principales.

Une relative au style général de l'application et qui est importée dans l'index principal et qui est héritée à toutes ses déclinaisons possibles.

Puis deux petites feuilles de styles indépendantes relatives aux formulaires d'inscription et d'authentification, nous servant à dispatcher le css entre plusieurs pages pour mieux s'y retrouver.

En ce qui concerne la charte graphique du site, nous avons opté pour des couleurs complémentaires nous garantissant une expérience UI optimale. Notamment pour favoriser et améliorer la lecture.

D'autre part, nous sommes passés par différentes maquettes graphiques Figma avant d'opter pour celle que nous présentons.

écran 1

écran 2

écran 3

écran 4

Structure des fichiers JS

Nous nous sommes servi du JS afin de créer des animations entre les pages. Plus précisément à l'ouverture de ces dernières en créant une reveal animation sur les différents éléments composant le main principal affiché.

Afin de créer une fluidité de navigation entre les pages, puis en instaurant des délais entre les différents éléments de la page dans le but de créer une hiérarchie entre les éléments affichés et donner un sens à la lecture globale de la page.

Merci de votre lecture et à bientôt sur FoodSens 😀