Quentin Leclaire

2024

Dossier projet

TP : Développeur Web et Web Mobile



Table des matières

[Compétences acquises 2](#_Toc167442616)

[Remerciements 3](#_Toc167442617)

[Introduction 4](#_Toc167442618)

[Présentation de l’entreprise 5](#_Toc167442619)

[Exercice pratique 6](#_Toc167442620)

[Projets 14](#_Toc167442621)

[Conclusion 15](#_Toc167442622)

# Compétences acquises

1. Symfony 6

* <https://symfonycasts.com/screencast/symfony6>
* <https://symfonycasts.com/screencast/symfony-fundamentals>
* <https://symfonycasts.com/screencast/symfony-doctrine>

1. API Platform

* <https://symfonycasts.com/screencast/rest>
* <https://symfonycasts.com/screencast/rest-ep2>

1. REACT

* <https://grafikart.fr/formations/react>

# Remerciements

Je tiens d’abord à remercier Camille LECLERCQ, conseillère à la mission locale, pour m’avoir dirigé vers cette formation.

Je voudrais également remercier ma formatrice Kesary PAVADAY pour la qualité de ses enseignements et son envie de partager ses connaissances.

J’étends aussi mes remerciements à l’équipe de gaea21 qui m’a accueilli dans leur organisation, et à mon tuteur de stage Alexandre Nguyen pour sa patience et ses explications.

Enfin, je remercie la région pour avoir financé cette formation.

# Introduction

J’ai intégré la formation Développeur Web et Web Mobile en septembre 2023 où j’ai pu développer mes compétences en HTML, CSS, PHP et JavaScript. Cette formation inclus également une période de 3 mois en entreprise que j’ai effectué avec gaea21.

Ce stage s’est déroulé du 19 février au 17 mai. Après une brève période d’intégration au sein de l’association, j’ai été formé au framework Symfony, à l’API Platform ainsi qu’à la libraire REACT. Cette formation s’est conclue par un exercice sous plusieurs variantes visant à démontrer ma compréhension de ces différents outils. J’ai ensuite été intégré à différents projets, mais ayant passé plus de temps que prévu sur les formations, je n’ai pas pu fournir un travail suffisant avant la fin de mon contrat de stage.

Au cours de ce rapport, je présenterai la version finale de l’exercice regroupant Symfony, une API et REACT. Je vais également faire une brève présentation des projets auxquels j’ai été assigné.

# Présentation de l’entreprise

Gaea21 est une association à but non-lucratif fondée en 2005 par Yvan Claude, son actuel président. Elle se situe à Genève, en Suisse. Elle œuvre pour la protection de l’environnement, la promotion du développement durable et la sensibilisation aux enjeux écologiques.

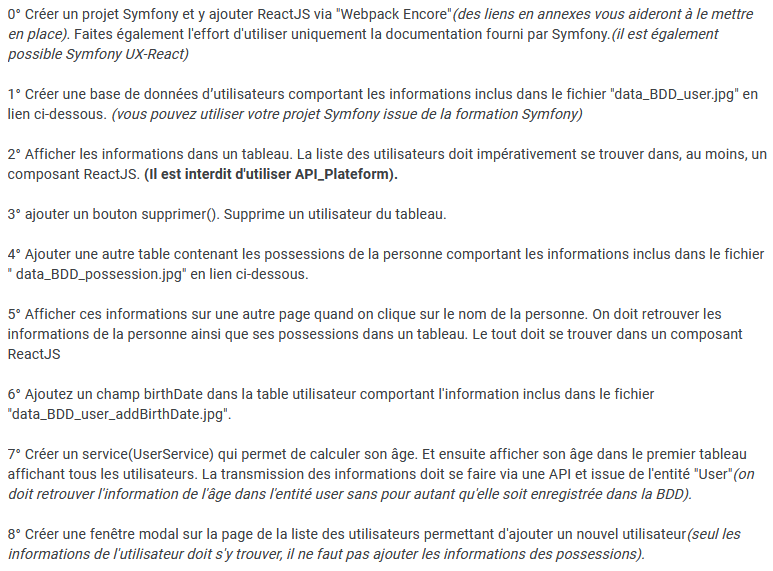
Gaea21 se divise en plusieurs projets, notamment :

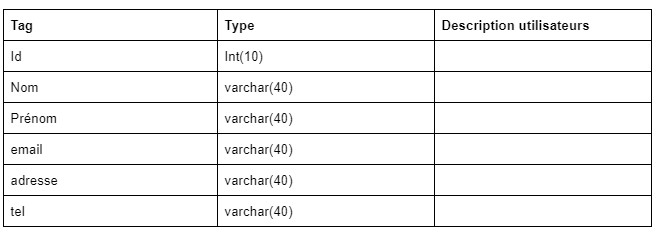
* Le Répertoire Vert, qui offre un référencement et évaluation de produits et services professionnels respectueux du développement durable, dans le but de diriger les consommateurs vers des entreprises vertes près de chez eux.
* Le Green Event Series, un ensemble d’évènements de sensibilisation au développement durable organisés par l’association, comme par exemple les Green Drink.
* Le Sustainable Living Program, programme visant à sensibiliser et valoriser le changement de comportement pour les individus.

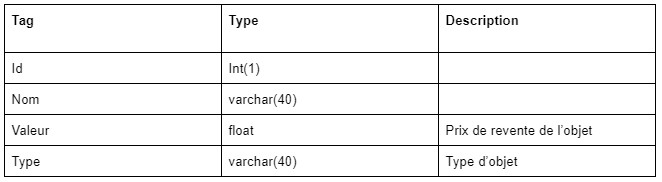
Le travail chez gaea21 est principalement réalisé à distance via le télétravail. Cela permet de limiter ses déplacements et donc ses émissions de carbone, en plus d’éliminer les restrictions géographiques qu’imposent le travail en présentiel. La communication se fait principalement via la suite Google (Gmail, Meet, Chat, ect…) mais également avec Skype.

# Exercice pratique

L’exercice de fin de formation consiste à créer un CRUD, d’abord sous Symfony, puis avec REACT, et enfin une combinaison des deux. La page d’accueil de cette application web doit afficher une liste d’utilisateurs avec leur nom, prénom, e-mail, adresse, numéro de téléphone et plus tard leur âge. On doit pouvoir supprimer un utilisateur de la liste et également en ajouter un via un formulaire. Enfin, on doit pouvoir cliquer sur le nom de la personne pour aller vers une autre page comprenant ses informations ainsi que ses possessions.







J’ai commencé par utiliser la commande ‘symfony new symfonyApp –webapp’ pour créer un nouveau projet symfony, puis j’installe Composer et les dépendances nécessaires, Node.js et Webpack Encore.

Composer est un géreur de dépendance PHP. Il est crucial dans la mise en place d’un projet Symfony car c’est grâce à lui qu’on télécharge et installe les dépendances, que ce soit pour un nouveau projet ou un projet existant.

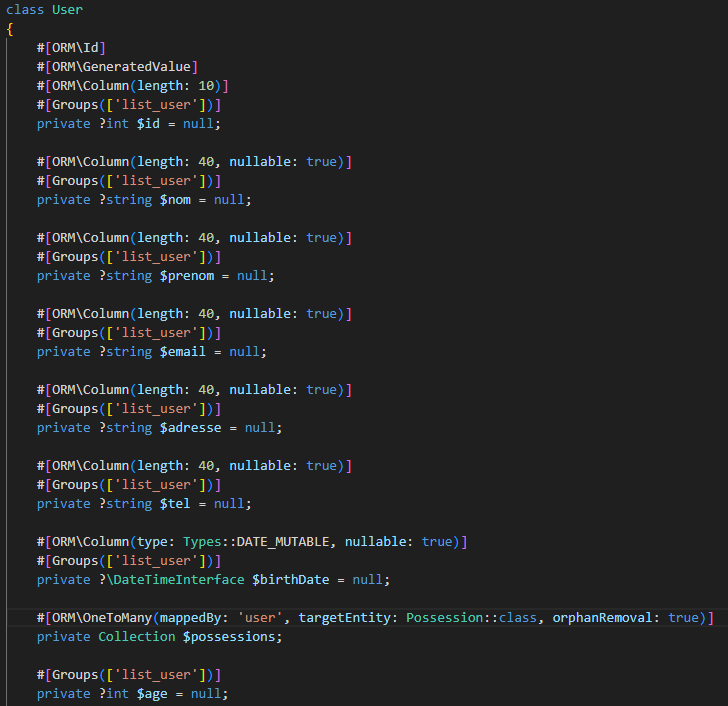
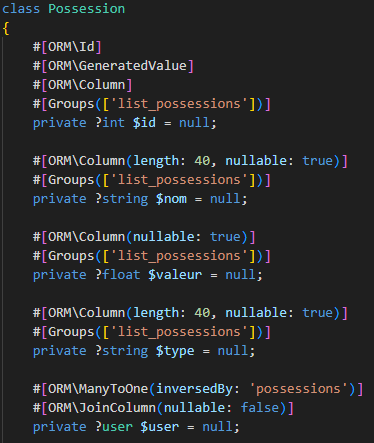
Node.js sert à utiliser du JavaScript sans passer par un navigateur web, et est ici utilisé pour son gestionnaire de paquets npm. npm a ici le même rôle que Composer, mais pour les projets JavaScript (REACT étant un framework JS).

Webpack Encore est un outil de bundling polyvalent largement utilisé dans le développement web moderne. Il optimise les fichiers source d’un projet et les regroupe dans des bundles prêts à l’emploi.

Ensuite, je modifie mon fichier .env pour y configurer l’URL de ma base de données. Je modifie l’adresse IP, le port, le nom de la BDD, les informations de connexion et la version du serveur.

Au lieu de créer ma base de données via des requêtes SQL, j’utilise un outil intégré à Symfony appelé Doctrine. Doctrine permet de créer des objets PHP appelés entités et va s’en servir pour effectuer les requêtes SQL. D’abord, je lance la commande ‘symfony doctrine:database:create’ pour créer ma BDD. Ensuite j’utilise la commande ‘symfony make:entity’ pour créer mon entité user et ses différentes données.

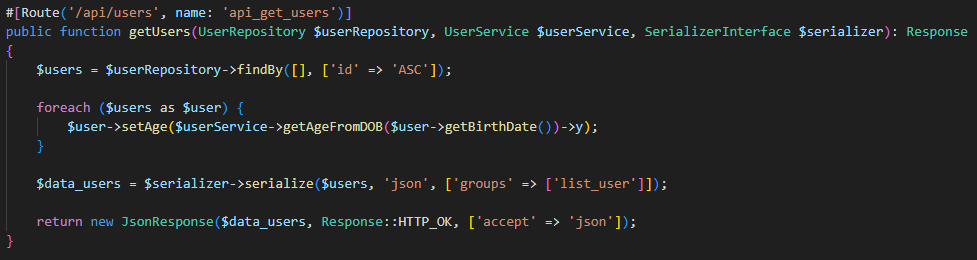
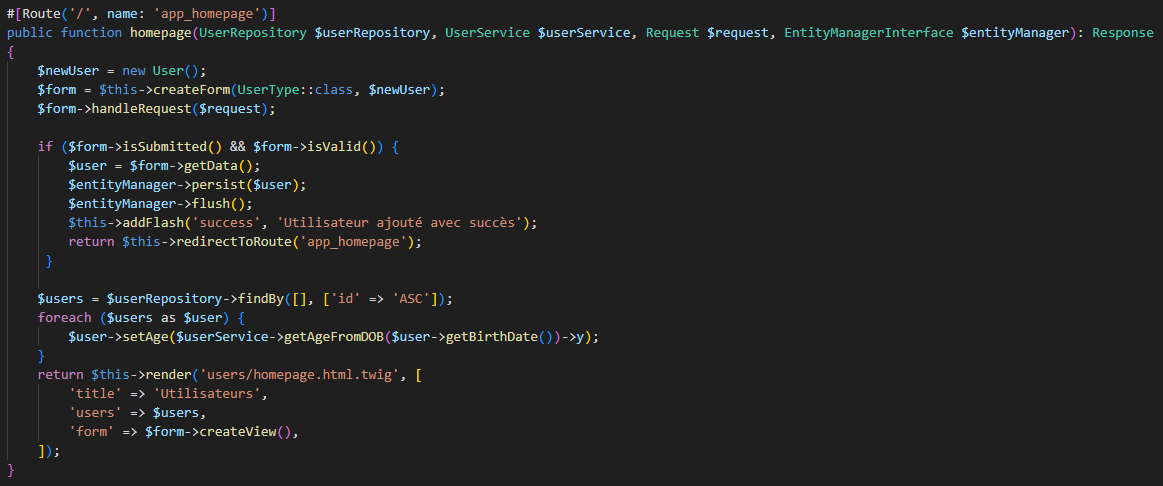
Pour les possessions, j’utilise des propriétés de type relation pour lier les utilisateurs à leurs possessions et vice-versa. Je fais également attention à ajouter les propriétés dans des groupes. Cela évite les erreurs de référence circulaire, où deux entités liées causent une boucle infinie de récupération de valeurs. Par exemple, en récupérant les informations d’un utilisateur, on récupère les informations de ses possessions, mais ces possessions ont également comme propriété les différents utilisateurs auxquels elles appartiennent, donc on doit récupérer les informations des utilisateurs, et cetera à l’infini.

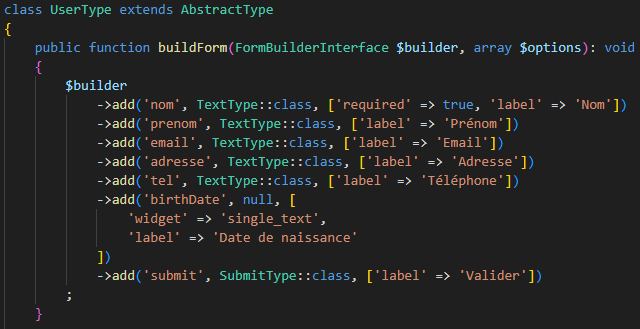


Enfin, j’utilise la commande ‘symfony doctrine:database:migrate’ pour effectuer une migration, c’est-à-dire la conversion des entités en requêtes SQL. Une fois ma BDD créée, j’utilise phpMyAdmin pour y inscrire des premières valeurs.

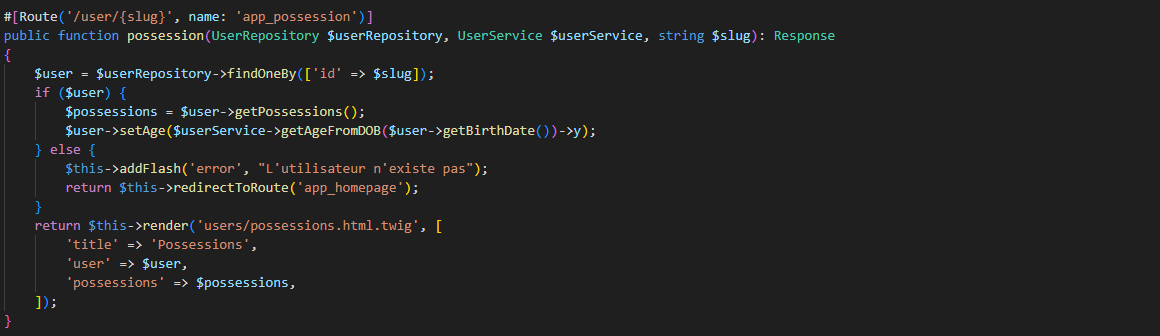
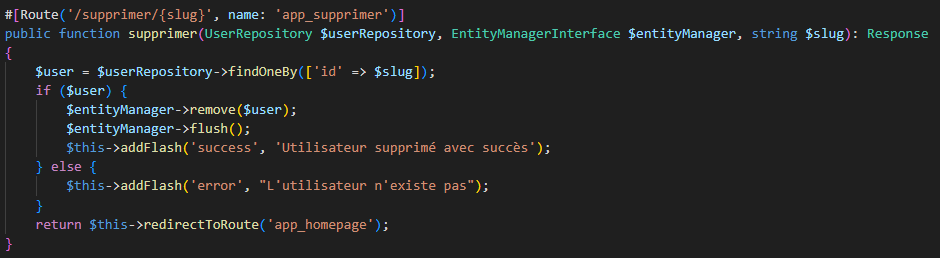
Maintenant que la BDD est créée, j’installe les composants UX-REACT et Serializer de Symfony. UX-REACT permet la communication entre REACT et Symfony, et Serializer permet de récupérer des données et les transformer en format JSON. J’installe également babel pour pouvoir convertir mes fichiers JSX en fichier JS utilisables par Symfony, et Bootstrap pour modifier l’apparence du site. Puis, j’ajoute la ligne suivante à ‘app.js’ et je décommente ‘.enableReactPreset()’ dans ‘webpack.config.js’.

La prochaine étape est de créer le contrôleur, j’utilise donc la commande ‘symfony console make:controller’. Je crée une route API pour récupérer les données via JSON, puis des routes correspondantes à la page d’accueil et la page d’informations et de possessions d’un utilisateur, et enfin des routes pour l’ajout et la suppression d’un utilisateur.

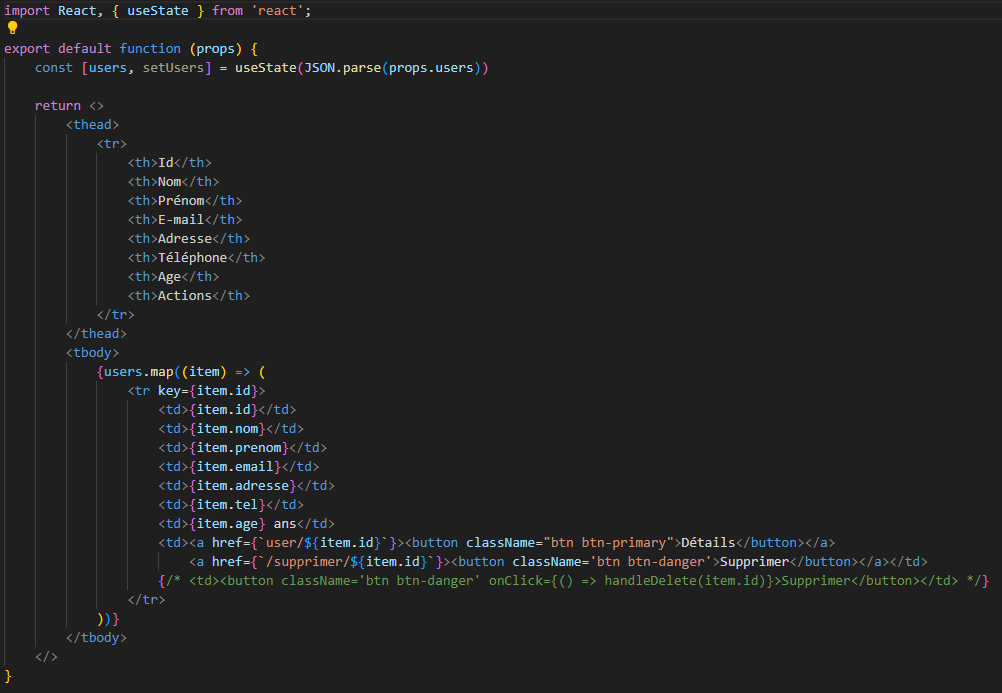
Dans cette route, je retourne une réponse JSON à partir des données récupérées de la base de données, la conversion étant faite par le Serializer. J’utilise également un service que j’ai créé qui sert à obtenir l’âge d’une personne à partir de sa date de naissance. Au final, je n’ai pas utilisé cette route dans mon application car les méthodes fournies par Symfony sont plus pratiques dans le cadre de cet exercice, mais j’ai compris l’intérêt de transmettre les informations de cette manière pour qu’elles soient utilisées par d’autres projets par exemple.

Cette route gère le rendu la page d’accueil de l’application. Je crée un formulaire grâce aux méthodes fournies par Symfony car il sera utilisé dans la fenêtre modale. J’ai également codé la validation du formulaire avec un message qui s’affiche pour l’utilisateur, et comme pour la route précédente, je récupère les données de la BDD avec conversion de la date de naissance en âge.

Voici la création d’un formulaire avec Symfony. Cela évite de devoir le créer manuellement dans le rendu de la page.

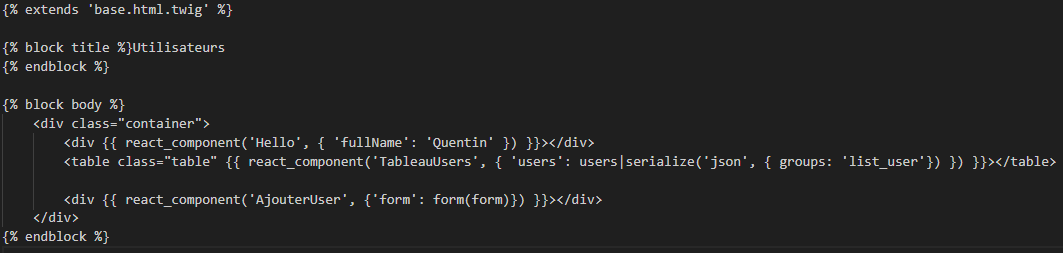
Cette route correspond à la page « Possessions » d’un utilisateur. ‘slug’ est un nom donné aux valeurs variables de l’URL, ici correspondant au numéro d’identification de l’utilisateur. Si l’identifiant ne correspond à aucun utilisateur, on retourne à la page d’accueil avec un message d’erreur.

Ceci est la route qui va gérer la suppression d’un utilisateur. Aucune page n’est rendue puisqu’on effectue une redirection à la fin de la procédure.

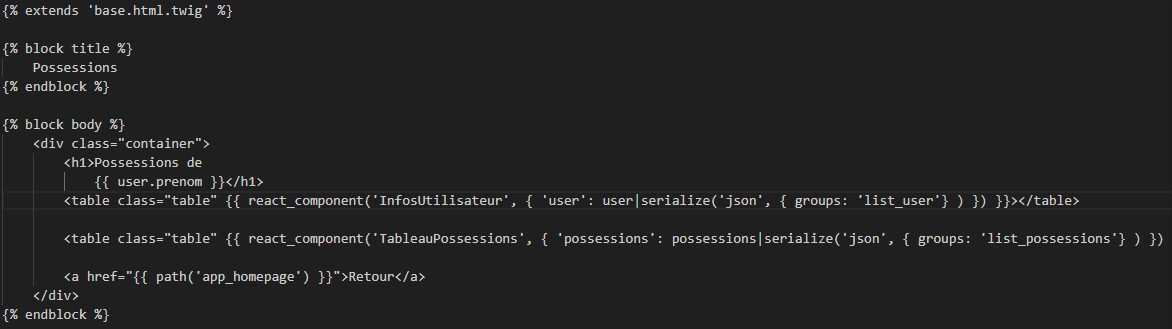
Je crée ensuite les différents composants REACT : le tableau des utilisateurs, le tableau avec les informations d’un utilisateur choisi, le tableau de ses possessions, et une fenêtre modale avec le formulaire d’ajout.

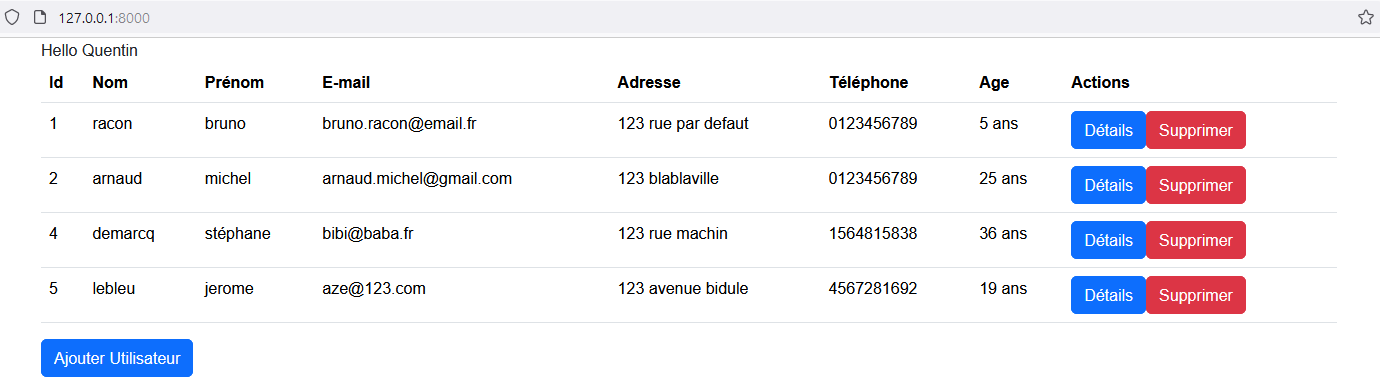


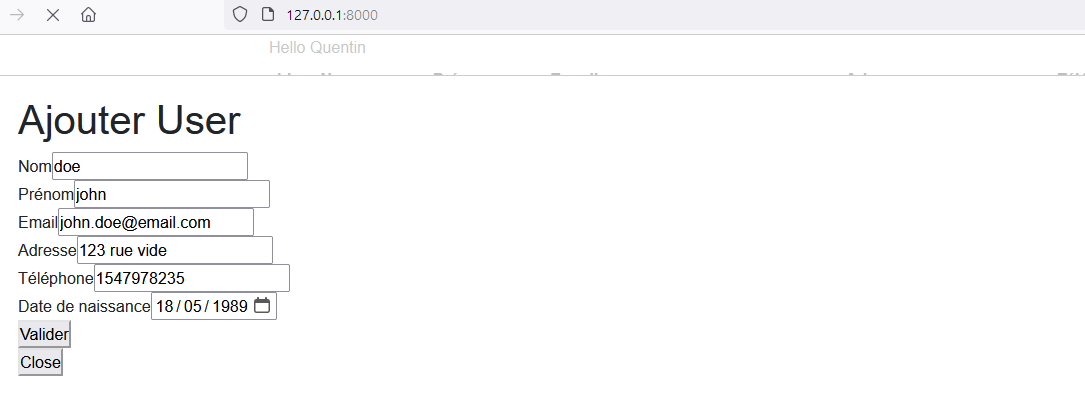


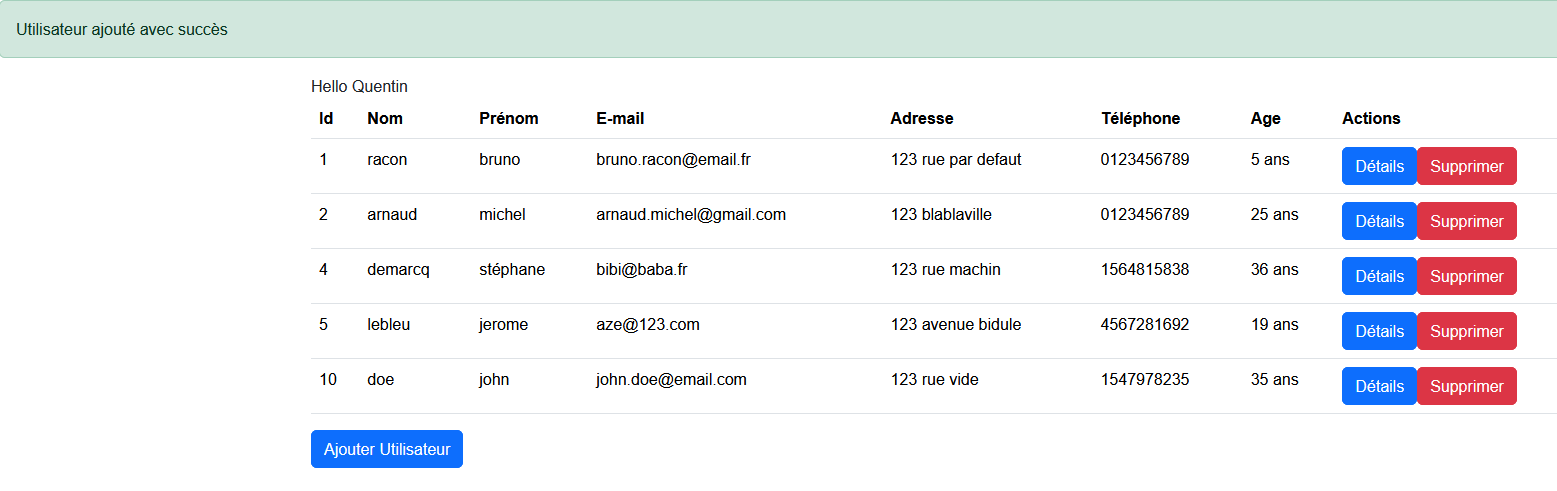
Enfin, j’appelle ces composants dans des fichiers Twig. Twig est le moteur de templates PHP par défaut par Symfony.

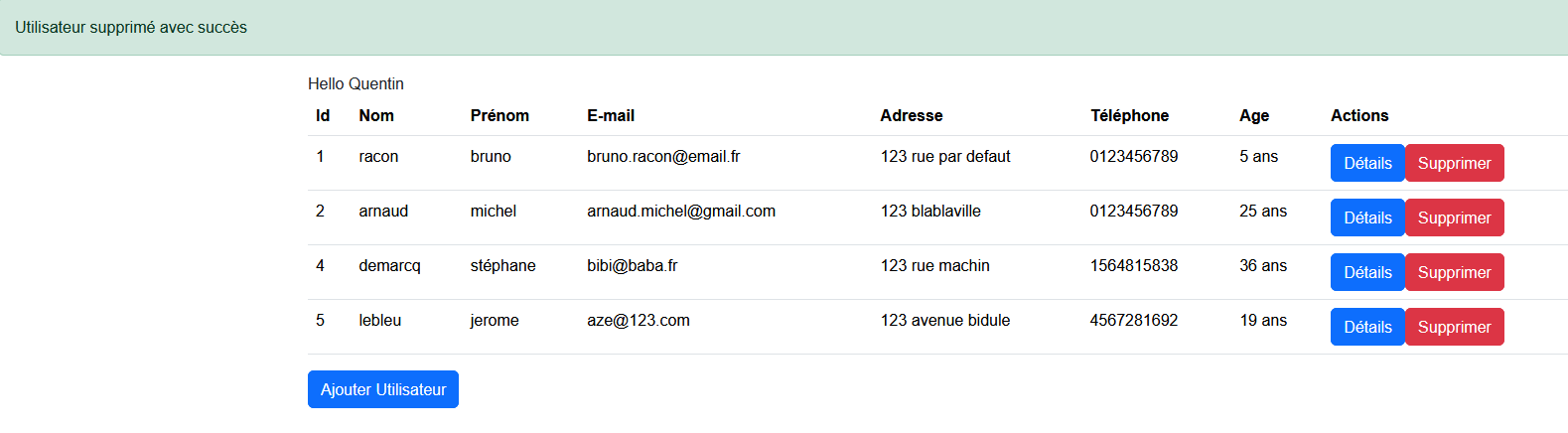
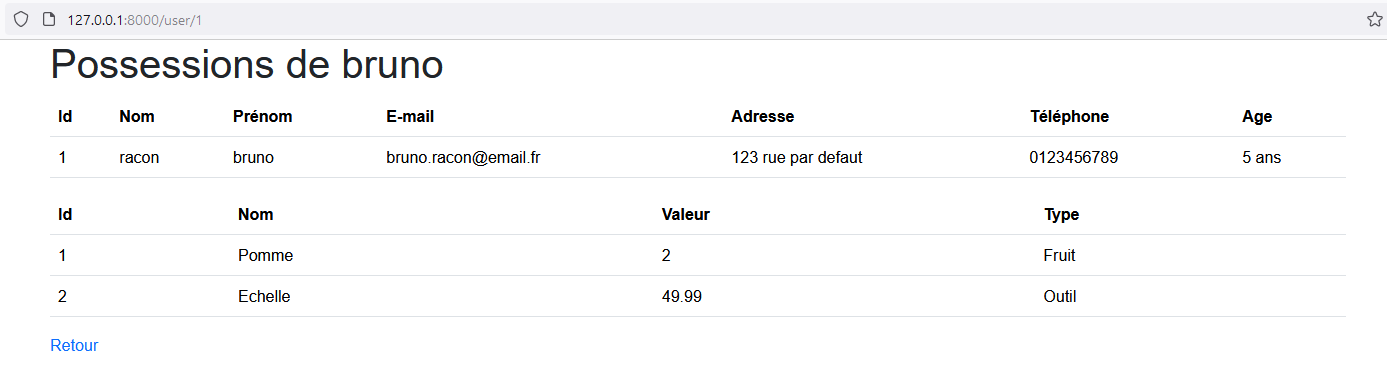
Voici le fichier de la page d’accueil. Ici j’appelle juste les deux composants REACT pour le tableau des utilisateurs et le modal d’ajout. J’utilise le Serializer pour convertir la liste des utilisateurs en objet JSON, en spécifiant le groupe de propriétés que j’ai défini dans la classe Utilisateur.

Pour la page de possessions, j’appelle là aussi les composants REACT correspondants avec les données converties en JSON.

Une fois terminé, je lance un serveur local avec la commande ‘symfony serve -d’. Je peux alors accéder à mon application via un navigateur web.







# Projets

Comme je l’ai dit dans l’introduction, mon temps passé en formation fut plus long que prévu et par conséquent j’ai été assigné que très tardivement aux projets. Ces projets sont :

* Le site web du Répertoire Vert.
* Le système d’authentification (gaeauser).
* Le service de traduction utilisés par les sites web de gaea21.

J’ai prévu de retourner chez gaea21 pour travailler sur ces projets.

# Conclusion

Pour conclure, je suis satisfais des compétences acquises chez gaea21. Je n’ai peut-être pas pu travailler sur un projet concret, mais j’ai pu néanmoins apprendre des choses qui n’ont pas pu être abordées dans la formation à l’AFCI. J’espère que le temps supplémentaire que je vais passer chez eux me permettront de mettre en œuvre ces appris et de fournir un travail à la hauteur de leurs attentes.