

Dossier Projet

HAMIDOU JIHANE

Titre professionnel visé

DEVELOPPEUR WEB ET WEB MOBILE

# Table des matières

[Remerciements 3](#_bookmark0)

1. [Liste des compétences du référentiel couvertes par le projet 4](#_bookmark1)
2. [Résumé du projet 5](#_bookmark2)
3. [Cahier des charges 6](#_bookmark3)

[Contexte 6](#_bookmark4)

[Objectifs 6](#_bookmark5)

[Spécifications fonctionnelles 8](#_bookmark6)

[Le contenu de l’application 8](#_bookmark7)

1. [Spécifications techniques du projet 10](#_bookmark8)

[Frontend 10](#_bookmark9)

[Backend 10](#_bookmark10)

[Sécurité de l’application 11](#_bookmark11)

[Base de données 12](#_bookmark12)

[Outils utilisés 13](#_bookmark13)

1. [Réalisations 14](#_bookmark14)

[Partie front-end 14](#_bookmark15)

[Maquetter une application 14](#_bookmark16)

[Réaliser une interface utilisateur web statique et adaptable 16](#_bookmark17)

[Développer une interface utilisateur dynamique 17](#_bookmark18)

[Partie back-end 19](#_bookmark19)

[Créer une base de données 19](#_bookmark20)

[Développer les composants d’accès aux données 22](#_bookmark21)

[Développer la partie back-end d’une application web ou web mobile 28](#_bookmark22)

1. [Présentation du jeu d’essai 30](#_bookmark23)
2. [Veille sur les vulnérabilités de sécurité 38](#_bookmark24)

[Validation de l’email et du mot de passe 38](#_bookmark25)

[Validation en PHP 38](#_bookmark26)

1. [Description d’une situation de travail ayant nécessité une recherche à partir de site](#_bookmark27) [anglophone 40](#_bookmark27)

[Texte original 40](#_bookmark28)

[Traduction 40](#_bookmark29)

1. [Conclusion 41](#_bookmark30)
2. [Annexes 42](#_bookmark31)

[Base de données 42](#_bookmark32)

[Dictionnaire de données 43](#_bookmark33)

[Routes (endpoints) 46](#_bookmark34)

# 1-Remerciements

J’aimerais tout d’abord remercier l’AFPA pour la formation proposée de Développeur web et web mobile, malgré un contexte parfois défavorable à un apprentissage sain, l’organisme m’a donné la chance d’apprendre et de m’améliorer dans des compétences importantes dans le développement web.

Merci au Pôle Emploi qui m’a soutenu financièrement dans ce projet professionnel.

Je souhaite remercier également l’entreprise Solutions909, qui m’a offert la chance de réaliser un stage dans le pôle développement de leur équipe, et particulièrement ma tutrice Emilie Rabito, qui m’a appris énormément de choses dans les domaines qu’elle maitrisait particulièrement bien. Une équipe de collaborateurs soudée, qui m’ont appris à travailler en équipe, à respecter les normes d’entreprise et à évoluer humainement et professionnellement dans ce sens-là.

A Pierre Denaës et Benjamin Gouget pour leur humanité, leur humilité, leur disponibilité, leur partage, leur gentillesse et pour avoir su trouver la force et la faculté de rattraper le de leurs prédécesseurs.

A mes camarades de promotion qui ont été là lorsque j’avais des questions, avec qui nous nous sommes entraidés, mes camarades de projet, avec qui j’ai collaboré pour réaliser nos premières applications et nos premiers sites.

Merci à tous.

# 

# 2-Présentation de l’entreprise où j’ai réalisé mon stage

Solutions909 est une entreprise à taille humaine qui œuvre pour la protection des données personnelles à travers le Règlement Général de Protection des Données (RGPD) et la cybersécurité. Son engagement vise un plus grand respect de la protection de la vie privée des individus.

Issus de l’édition de logiciel dans le domaine de la santé, ainsi qu’une de plusieurs années dans l’accompagnement des organismes de soin (centres hospitaliers, cliniques, cabinets de ville ou ESMS). Les données de santé sont des données sensibles, une attention toute particulière doit être apportée à leur traitement.

* 1. **Résumé des projets**

Dans le cadre de ma formation de développeur web et web mobile à l’AFPA saint jerome ,

j’ai réalisé un stage dans l’entreprise Solutions 909 .

Durant la période de stage j’ai travaillé sur 2 sites ;

* Projet Solutions909 : Il a été réalisé avec le CMS Bolt qui est fait en Symfony (Framework PHP). Le but de ce projet était de rendre facile et adaptable le contenu du site (comme un site wordpress) mais en ayant un accès total au code et en créant notre propre thème.
* Gigisbrand : Il a été créé avec Symfony (Framework PHP) dont l’approche se base sur des composants et du modèle MVC (Model-View-Controller). Un site e-commerce de vente de vêtements femme .

1. **CMS** : « Content Management system » Il s’agit d’un programme informatique utilisant une base de données et permettant de gérer de A et Z l’apparence et le contenu d’un site web. Il permet, à des individus ou à des groupes hiérarchisés, de mettre à jour le contenu d’un site web à partir d’un panneau d’administration.
2. **Symfony** : Symfony est un ensemble de composants PHP ainsi qu'un framework MVC libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d’accélérer le développement d'un site web.

# Projet solutions909

# Cahier des charges

### Le cahier des charges définit la relation avec le client. Il permet de lister les différentes fonctions attendues, de mettre en place une charte graphique, ainsi que le coût de revient et les délais imposer.

### Contexte

Solutions 909 est une société qui aide les entreprises à se mettre en conformité avec le RGPD3. A mon arrivée dans l’entreprise on a entrepris ensemble la création du cahier des charges . Le service marketing avait fait des rapports concernant les types de pages et leurs contenus vu qu’il en avaient déjà un site , le graphiste en charge du design a créé les logos , on s’est mise d’accord sur les couleurs .

### Objectifs

L’objectif de ce site est de mettre en avant les produits commercialisé par l’entreprise , ainsi que sensibiliser les utilisateurs sur l’importance de la protection des données personnelles et surtout les données de santé sur la partie BLOG4 du site .

Il doit être 100% administrable et accessible depuis tous les navigateurs modernes (Chrome, Edge, Firefox, Opéra et Safari) et depuis différents appareils (ordinateurs de bureau, tablettes et smartphones).

### Budget

Sans objet car c’est pour l’entreprise où j’ai réalisé mon stage .

### Délai

Mercredi 5 Octobre 2022 .

### Charte graphique

Le site général , ainsi que le blog doivent avoir les mêmes couleurs et polices . Le bleu foncé sera la couleur dominante, accompagné de ses dégradés. La polices de caractères utilisée sera ‘Poppins’ pour tout le site .

3 **RGPD** : Le règlement général de protection des données (RGPD) est un texte réglementaire européen

qui encadre le traitement des données de manière égalitaire sur tout le territoire de l’Union Européenne. Il est entré en application le 25 mai 2018*.* - *Source economie.gouv.fr*

4 **Blog** : Un blog est un type de [site web](https://fr.wikipedia.org/wiki/Site_web)  ou une partie d'un site web utilisé pour la [publication](https://fr.wikipedia.org/wiki/Publication) périodique et régulière d'articles personnels, généralement succincts, rendant compte d'une actualité autour d'une thématique particulière.

# METHODE AGILE

C’est la présentation hebdomadaire de l’avancement du projet au client sous forme de sprint.

C’est également l’organisation des différentes tâches à réaliser en sous-tâches.

Pour faciliter mon organisation j’ai utilisé la suite Atlassian (Trello, Jira, Confluence) en ligne pour la gestion des tâches à accomplir, et aussi Git et GitHub pour le versioning.

# 

5 **Bootstrap** : Bootstrap.css est un framework CSS qui organise et gère la mise en page d'un site web

# ARBORESCENCE DU SITE

|  |  |
| --- | --- |
| **Pages** | **Détails** |
| **Page d’Accueil** | Landing page du site . |
| **Page Audit** | Information sur les différents audits proposé par l’entreprise . |
| **Page Accompagnement** | Différents type d’accompagnements. |
| **Page Formation & sensibilisation** | Les formations proposées par l’entreprise. |
| **Page Santé** | Protection des données de santés |
| **Page Legal tech** | Information superDPOS |
| **Page Cybersécurité** | Les offres cybersécurité |
| **Page de Contact** | Formulaire de contact |
| **Page Blog** | Pour accéder au blog solution909 |
| **Page Mentions**  **légales** | Les mentions l’gales |
| **Page Mentions**  **RGPD** | Les mentions RGPD |
| **Pages d’erreur** | Les pages 404 ,403 ,500 ,503 |
|  |  |

### Spécifications fonctionnelles

Le dirigeant voulais que le site soit fait avec BOLT cms . Ayant appris à utiliser Bootstrap5 pendant ma formation , j’ai opté pour ce framework CSS pour la partie front .

# Spécifications techniques du projet

### Frontend

La partie front-end de mon site a été réalisée avec Boltcms.

C’est un système de gestion de contenu gratuit basé sur PHP. Il est sorti en 2012 et développé par Two Kings et la communauté Bolt.

Bolt utilise Twig6 pour les modèles et inclut des fonctionnalités pour la gestion du contenu et des utilisateurs. Il est open-source, très pratique car l’interface Admin est déjà faite , in suffit de créer le projet , lancer une commande pour la création de la base de données et créer des contenue pour l’affichage, il permet aussi de créer son propre theme.

### Backend

La partie back-end a été réalisée à l’aide du framework PHP **Symfony** dont l’approche se

base sur des composants ainsi que du modèle MVC (Model-View-Controller).

BoltCMS génère sa propre administration du site (les bases) mais j’ai réalisé toute la personnalisation de la partie d’administration, et j’ai aussi intégré un formulaire de contact via SendInBlue qui est un service de gestion de newsletter et de liste de contact (déjà utilisé par l’entreprise qui souhaitait centraliser au même endroit tous ses contacts).

J’ai modifier les fichiers de configuration pour les rendre compatibles avec nos attentes et créé des types de contenu (ContentType) et facilité leur administration.

6**Twig** : Twig est un moteur de templates pour le langage de programmation PHP, utilisé par défaut par le framework Symfony.

# Projet Gigisbrand

**Contexte du Projet**

Gigisbrand est un projet réalisé avec Symfony , j’ai commencé a travaillé dessus avant la fin de formation .

Lors de mon entretien pour effectuer mon stage , le dirigeant de l’entreprise m’a informé qu’ils travaillent avec Boltcms pour la partie front-end , et Symfony pour le back-end, et vu que j’avais un mois avant de débuter le stage, j’ai décidé de réalisé un site e-commerce avec ce framework .

### Objectifs

Un site de vente pour femme .

**Contenus sur le site**

Le site contiens des produits de différentes catégories se compose de 17 pages .

# ARBORESCENCE DU SITE

|  |  |
| --- | --- |
| **Pages** | **Détails** |
| **Page Home** | Landing page du site .  Mise en avant des produits . |
| **Page Category** | Affiche les articles de chaque catégorie . |
| **Page Product** | Affiche un seul produit. |
| **Page Cart** | Affiche les produits ajouté dans le panier avec le prix totale et le nombre d’article . |
| **Page Purchase confirm** | Affiche le formulaire de confirmation de commande . |
| **Page Login** | Formulaire de connexion |
| **Page Register** | Formulaire d’inscription |
| **Page Profil** | Affiche le profil de l’utilisateur , c’est là ou il paut changer son mot de passe et voir la liste de ses commandes . |
| **Page Purchase** | Affiche l’historique de commande . |
| **Page Password change** | Affiche le formulaire pour changer le mot de passe |
| **Page Product edit** | Affiche le formulaire d’édition d’un produit |
| **Pages Product create** | Affiche le formulaire de création de produit |
| **Page Category edit** | Affiche le formulaire d’edition d’une catégorie |
| **Page Category create** | Affiche le formulaire de création d’une catégorie |
| **Page Admin** | Affiche l’interface admin , ou l’utilisateur peut créer et gérer des produits , catégorie et aussi les utilisateurs . |
| **Page Payment** | Affiche le formulaire de payement |
| **Page Logout** | Page de déconnexion |

### Fonctionnalités

### L’utilisateur peut s’inscrire , se connecter et se déconnecter , il aura aussi accès a son profil pour modifier son mot de passe et voir la liste de ses commande , ainsi qu’ajouter un produit au panier et confirmer sa commande jusqu’au formulaire de payement .

### L’admin aura accès a toute mes pages que l’utilisateur , et aussi a son interface de gestion pour gérer les produits , catégorie , commandes et utilisateurs .

# Spécifications techniques du projet

Le projet a été créer entièrement avec Symfony (frontend et backend) .

**Les bundles Symfony** :

Le bundle est une sorte de plug-in ou module permettant d’implémenter plusieurs fonctionnalités dans nos projets. En PHP il est considéré comme un namespace. Les bundles sont capables de modifier le comportement de notre application.

Un des gros avantages est qu’ils sont open source, totalement réutilisables dans la plus part de nos projets. Sur Symfony il existe des milliers de bundles et même des sites comme KpnBundle qui vous proposent de les télécharger. Encore une fois le principe du bundle est de ne pas réinventer la roue et de nous procurer un gain de temps considérable dans les phases de développement de nos sites. En plus d’être réutilisable les bundles sont flexibles mais surtout sécurisés. Par exemple EasyAdminBundle est un puissant outil d’administration de site sous Symfony. Dans nos projets avons fait appel à lui mais aussi à Webpack-Encore-Bundle.

**Webpack-Encore-Bundle**:

Webpack-Encore-Bundle a été le premier bundle installé par notre équipe. Ce bundle est une API (interface de programmation d’application). Il regroupe et gère la fonction de plusieurs modules JavaScript. C’est une bibliothèque Javascript qui va nous permettre le pré-processing Css et Js ainsi que la compilation et minification de ceci. Cette API nous permet de manager notre Css et notre Js. En vue de la customisation de Bootstrap, directement implanté dans Symfony ce bundle est apparu comme une évidence. La configuration Css de Bootstrap est écrite en SASS ou SCSS. Afin de modifier celle-ci nous avons besoin d’un pré-processeur css. De plus l’utilisation d’un préprocesseur Css coté Front-End permet un meilleur agencement et donne une logique au code design ainsi que des fonctionnalités.

Une fois le bundle installé celui-ci est fourni avec un fichier de configuration «  webpack.config.js  » qui nous servira à activer les différentes fonctionnalités ou 25 méthodes. Dans notre cas nous activerons le module « sass-loader » pour l’ensemble de nos projets. Une fois la gestion du module pris en charge par notre application, nous avons dû installer l’ensemble du Webpack-Encore ainsi que le module du Sass-loader avec Yarn, un gestionnaire de package JavaScript. Yarn nous servira également à exécuter les commandes liées à la réalisation du pré-processing.

\_**EasyAdminBundle.**

EasyAdminBundle est un bundle mis en avant par Symfony. C’est un bundle d’administration de nos « Entity » créant un BackOffice pour nos sites Web. Ce backOffice est livré avec une gestion du CRUD (Create, Read, Update, Delete), une fonction de recherche, une pagination et des templates responsives, le tout, clé en main ! Une fois le bundle installé, la création d’une première vue du BackOffice ne prendra que quelques secondes. Toute la configuration d’EasyAdmin se réalisera via un fichier «  .YAML  ». Cette configuration est relativement verbeuse mais considérée  comme «  human-friendly  ». Un des gros atout d’EasyAdmin est qu’il repose sur des concepts standards. Par exemple, pour ajouter une fonctionnalité de type CMS, gestion de média, il suffit de créer un contrôleur et de passer sur du code Symfony. Ce bundle est très simple à mettre en place et il **est** possible de développer et d’intégrer des fonctionnalités avancées en son sein. EasyAdmin, étant poussé par Symfony et allant exactement dans le sens de nos besoins c’est lui que nous choisirons. De plus ce bundle peut se compléter avec d’autres composants eux aussi sous forme de bundle. VichUploader est l’un des composants que nous avons ajouté au fonctionnement d’EasyAdmin.

**\_SecurityBundle**

Le bundle SecurityBundle fournit des fonctionnalités de hachage pour les mots de passe.

**Un gestionnaire de package.**

**\_Yarn.**

Yarn est un logiciel Open Source. Il permet la simplification de la gestion de dépendance dans nos projets. Yarn est un gestionnaire de package JavaScript pour NodeJs développé par Facebook et visant à améliorer les lacunes de « npm ». Yarn est compatible avec une grosse quantité d’infrastructures actuelles. Yarn est compatible avec « npm » et « bower » ce qui permet de limiter le nombre de gestionnaires de packages par projet. Pour éviter les problèmes de version de package, Yarn met à jour un fichier « yarn.lock » avec des numéros de versions fixes. Cela permet d’installer, d’une machine à l’autre exactement les bonnes versions de tous nos packages. Un des gros avantages de Yarn est qu’il entrepose les dépendances en cache ce qui permet d’éviter un retéléchargement de celle-ci à chaque nouveau projet. Yarn nous permettra d’installer et d’exécuter toutes les dépendances de notre projet coté Front-End. Par exemple, pour nos projets Symfony, nous avons inastallé via composer le bundle Webpack-Encore. Ce bundle intègre et configure Webpack-encore dans nos projets ensuite il faudra installer et exécuter la bibliothèque via le gestionnaire Yarn. En plus d’améliorer « npm » Yarn assure une sécurité absolue des dépendances qu’il gère. Yarn est un choix nécessaire dans notre volonté d’utiliser le format Scss pour la conception du design des deux sites web.

**L’ORM Doctrine** a été utilisé pour la gestion de base de données, en tant qu’ORM8 son rôle est de transformer les objets PHP en écriture SQL, et récupérer les écritures SQL et les transformer en objet PHP.

L’utilisation de Doctrine apporte une couche d’abstraction qui permet de se

concentrer sur la gestion de la logique métier du site. De plus, pour la sécurité du site il apporte nativement un ensemble de bonnes pratiques qui permettent entre autres d’éviter toutes les failles de type injection SQL .

Les **annotations** (de doctrine) servent à apporter de l’information. Par exemple les annotations Doctrine permettent de définir le type de colonne qu’il faut créer dans la table pour correspondre à un attribut de classe. Doctrine vient lire l’information contenue dans l’annotation, et agit en conséquence.

### Sécurité

Voters / hashpassword/contraints formulaire /securitybundel

### Base de données

Dans Gigisbrang , les tables qui sont principalement utilisées sont les suivantes :

|  |  |
| --- | --- |
| [**user**](http://localhost/phpmyadmin/index.php?route=/sql&server=1&db=gigisbrand&table=user&pos=0) | nom, code client, SIRET, adresse, adresse email, dates de création et de modification… |
| [**category**](http://localhost/phpmyadmin/index.php?route=/sql&server=1&db=gigisbrand&table=category&pos=0) | représentant légal du client : nom, prénom, SIRET,  personne physique ou morale, dates de création et de modification… |
| **Product** | contacts du client : nom, prénom, téléphone, adresse email, dates de création et de modification… |
| **Purchase** | documents du client : nom de document, extension, taille, dates de création et de modification… |

### Outils utilisés Gestion de projet

La suite Atlassian (Trello, Jira, et Confluence) nous a permis de gérer toutes nos tâches, en planifiant et ayant un suivi de chacune ainsi que du travail de toute l’équipe et de créer de la documentation où toute l’équipe peut collaborer.

### Versioning

Git, l’outil de versioning open-source par excellence. Utilisé tout au long de notre formation.

Nous avons utilisé la version GitHub Desktop pour plus de praticité du partage de code et publication du projet.

### Développement

VS Code, éditeur de code open-source. Très complet, utile tout au long de notre formation et customisable à l’aide de ses nombreuses extensions gratuites communautaires pour augmenter sa productivité.

### Base de données

PhpMyAdmin, logiciel libre écrit en PHP qui a pour mission de s’occuper de l’administration d’un serveur de base de données MySQL ou MariaDB. Très utile pour réaliser la plupart des tâches administration et exécution de demandes.

### Figma

Figma est un outil de tableau blanc collaboratif. Il permet en quelques secondes de commencer à dessiner, partager des idées ou animer une rétrospective. Ça m a permis de maquetter quelques pages rapidement.

# Réalisations

# Partie front-end

### Maquetter une application

### 

## Réaliser une interface utilisateur web statique et adaptable

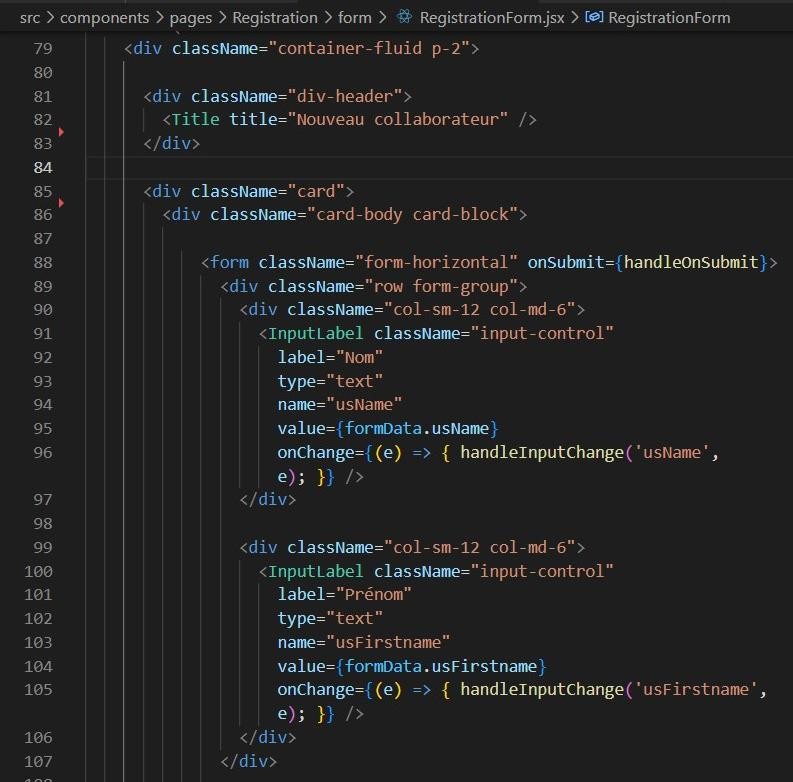
L’application ayant été composée **en mobile-first**, il a fallu adopter son affichage selon les supports utilisés (téléphone, ordinateur...) afin d’optimiser l’affichage, ceci a été fait grâce à l’utilisation de **Bootstrap**.

Bootstrap c’est un framework open-source créé par Twitter qui permet de créer des applications web qui s’adaptent aux écrans des PC, tablettes et smartphones grâce à son système de grille flexible et extensible. C’est ce que l’on appelle le responsive. Il est codé avec 3 langages de programmation, HTML5, CSS3 et JavaScript.

Le principe est simple, il s’agit de boite, un container principal, des row (ligne), des

colonnes et chaque colonne est divisée en 12.

Voici un extrait de code de l’utilisation de Bootstrap :



Extrait de code du RegistrationForm concernant l’affichage des champs du formulaire

### Développer une interface utilisateur dynamique

Lorsque nous sommes dans l’annuaire, l’utilisateur qui a crée un user peut l’éditer, mais seulement les informations générales. A savoir le login, l’email son groupe ainsi que les dates ou il aura accès à son compte. Ce système est prévu pour donner des droits temporaires (exemple un stagiaire a qui on bloquerait les accès lorsque son stage est terminé).

Le bouton sauvegarde met en base de données la modification et nous renvoit vers

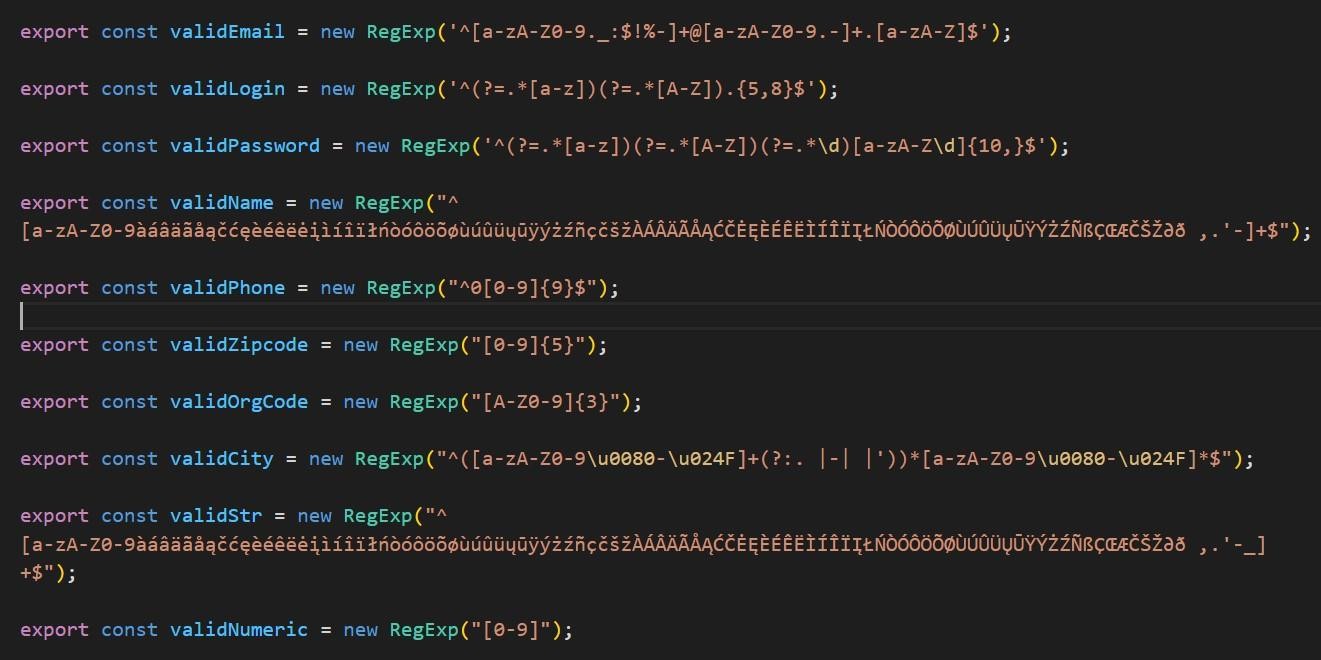
l’annuaire.



Edition d’un utilisateur

Une fois soumis, la fonction **HandleOnSubmit** est appelée et comprend les données du formulaire renseigné. Ensuite dans la fonction **updateUser**, présente dans le fichier **UserService.jsx** du dossier **services/API** contenant toutes les fonctions pour appeler nos endpoints associés aux utilisateurs (login, forgotPassword, register, getAllUsers…).





Cette fonction se chargera de passer ces valeurs dans la requête vers l’API. Pendant ce

temps on affiche un chargement sur le bouton sauvegarder, et une fois la requête à l’API

effectué la fenêtre se ferme automatiquement et un message de type toast s’affiche à

l’écran pour indiquer le succès de l’opération à l’utilisateur.

Si le formulaire est incomplet ou contient des erreurs, grâce à l’écoute et enregistrement des erreurs passées via la fonction **validateData** sur les composants input du formulaire, au submit des données les erreurs déclencheront l’affichage d’un message d’erreur sur les champs concernés et l’appel vers **udpateUser** ne sera jamais effectué.

### Partie back-end

**Créer une base de données**

Dans notre projet, nous avons utilisé une **base de données MySQL**. C’est une base de

données relationnelle qui correspond aux attentes et aux spécificités de notre projet.

De plus, les clés étrangères sont appliquées au niveau de la base rendant leurs inexistences impossible.

Dans notre cas, nous allons utiliser du multibase de données car nous avons besoin de créer une première base de données Admin\_Machine et une seconde Crypto.

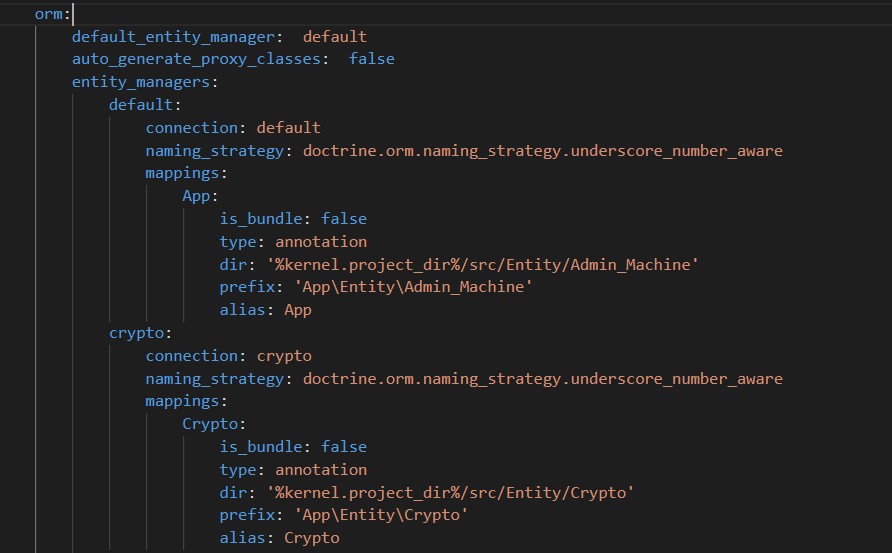
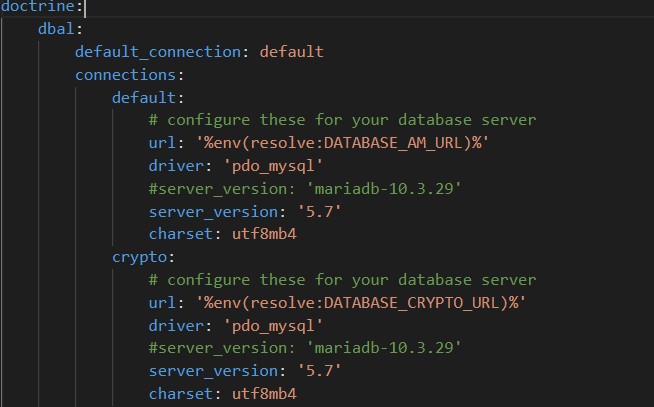
Pour cela, le Framework Symfony nous met à disposition une commande de terminal par l'intermédiaire de l'extension **Doctrine.**

Il est nécessaire de configuration Doctrine ORM dans Symfony et les informations de connexion s'inscrivent dans le fichier **.env** situé à la racine du projet.



Pour interagir avec la base de données depuis PHP, nous allons nous appuyer sur [Doctrine,](https://www.doctrine-project.org/) un ensemble de bibliothèques qui nous aide à gérer les bases de données : **Doctrine DBAL** (une couche d'abstraction de la base de données), Doctrine ORM (une librairie pour manipuler le contenu de notre base de données en utilisant des objets PHP), et **Doctrine Migrations.**

Dans le fichier de configuration doctrine.yaml, on configure de cette façon :



L'Entity Manager fait donc le lien entre les "Entités", qui sont de simples objets PHP, et la base de données.

Via **PhpMyAdmin**, nous avons créé les deux bases de données avec un interclassement (collation) « utf8mb4\_general\_ci » pour la prise en charge d’une large palette de caractères dont les caractères français et des émojis/smileys.

Pour identifier correctement le contenu de chaque base de données, dans les répertoires de Symfony (Controller, Entity et Repository), nous avons créé un répertoire par base de données pour question de lisibilité de code.

Nous avons utilisé la commande suivante **make :entity** pour générer nos classes

d’entités.

Voici un exemple de commande pour la création de la table UserKeys dans la base de données Crypto :



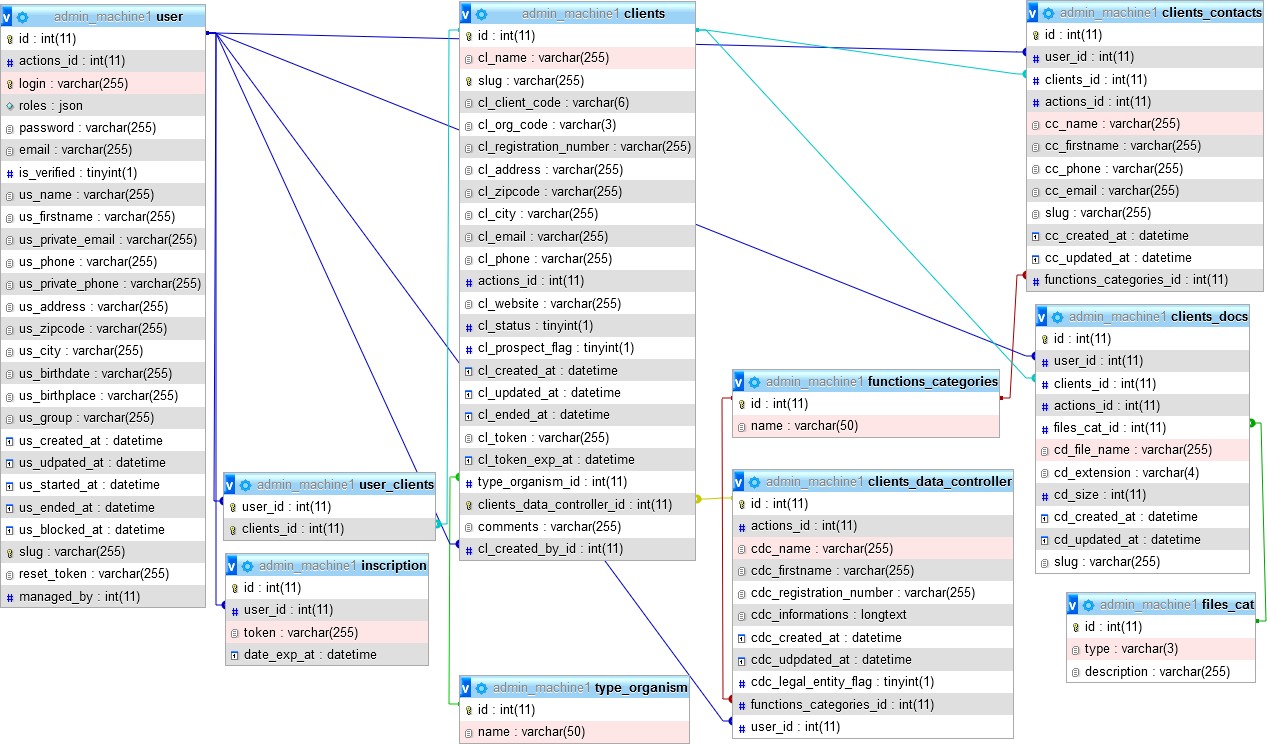
Puis une fois, les champs ajoutés dans l’entité, il faut mettre jour la base de données par

une migration en précisant quel gestionnaire d’entités est utilisé.

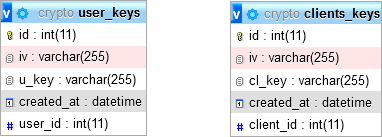


puis,





Représentation de la BDD Admin\_Machine



Représentation de la BDD Crypto

### Développer les composants d’accès aux données

Pour accéder à nos données, nous avons créé des classes entités (Entity) contenant un objet propre à ce que l’on souhaite représenté. Cette entité est ensuite connectée à une table de la base de données via l’ORM Doctrine.

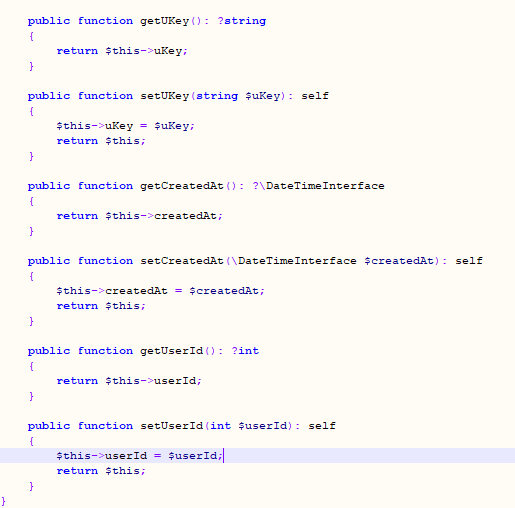
L’outil Doctrine de Symfony a été très utile pour modéliser rapidement des entités via la commande **php bin/console make:entity** celui-ci nous pose alors des questions sur le nom de l’entité désirée, ainsi que le nom de chaque propriété à ajouter, son type et sa longueur.

Une fois toutes les propriétés énumérées, le fichier contenant notre entité est créé et est disponible dans le dossier des entités. Ce fichier contient les propriétés déclarées ainsi que leurs getters/setters et les méthodes obligatoires à implémenter. Se créé aussi le fichier repository contenant une classe de repository.

Le fichier créé est accessible à l’adresse **src/Entity/Crypto/UserKeys.php.**

*Figure 1 Extrait de l'entité UserKeys générée*





A ce stade l’entité est créée, mais n’existe pas dans la base de données. Il a fallu créer un fichier de migration qui va contenir le code SQL à exécuter dans notre gestionnaire de base de données (PhpMyAdmin). Ceci s’est fait via la commande **php bin/ console make:migration** dans le terminal.

Puis, la mise à jour la base de données en fonction du fichier précédemment généré.

Toujours en ligne de commande à l’aide de **php bin/console doctrine:migrations:migrate.**

À ce stade, les opérations sur la base de données (lecture, écriture, modification) peuvent

se faire sans saisir de requêtes via des méthodes en initialisant l’entité créée.

Toutefois il faut des méthodes pour se faire, nous avons créée des controllers, autrement appelée plus communément « action » placée dans le dossier **src/Controller**.

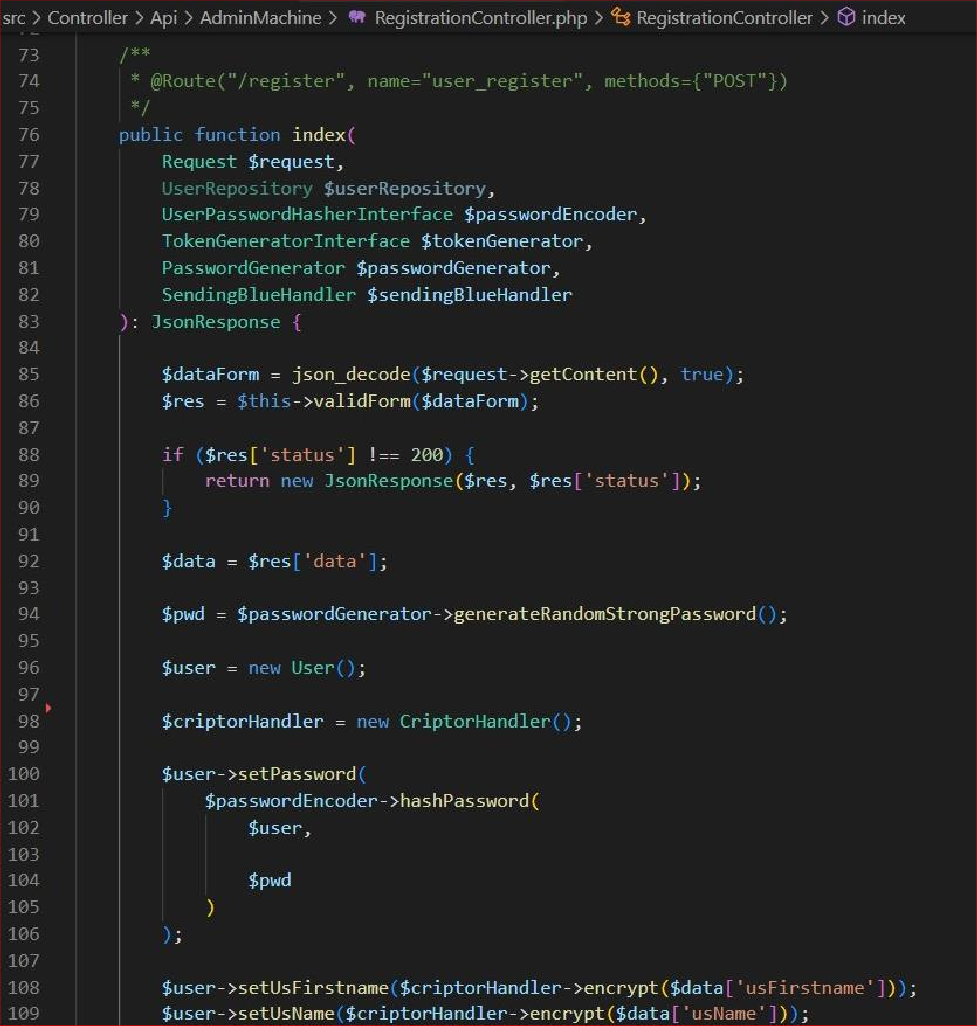
Le controller permet le lien entre la requête effectuée, l’action qui en découle et l’opération sur une entité puis le lien en base de données.

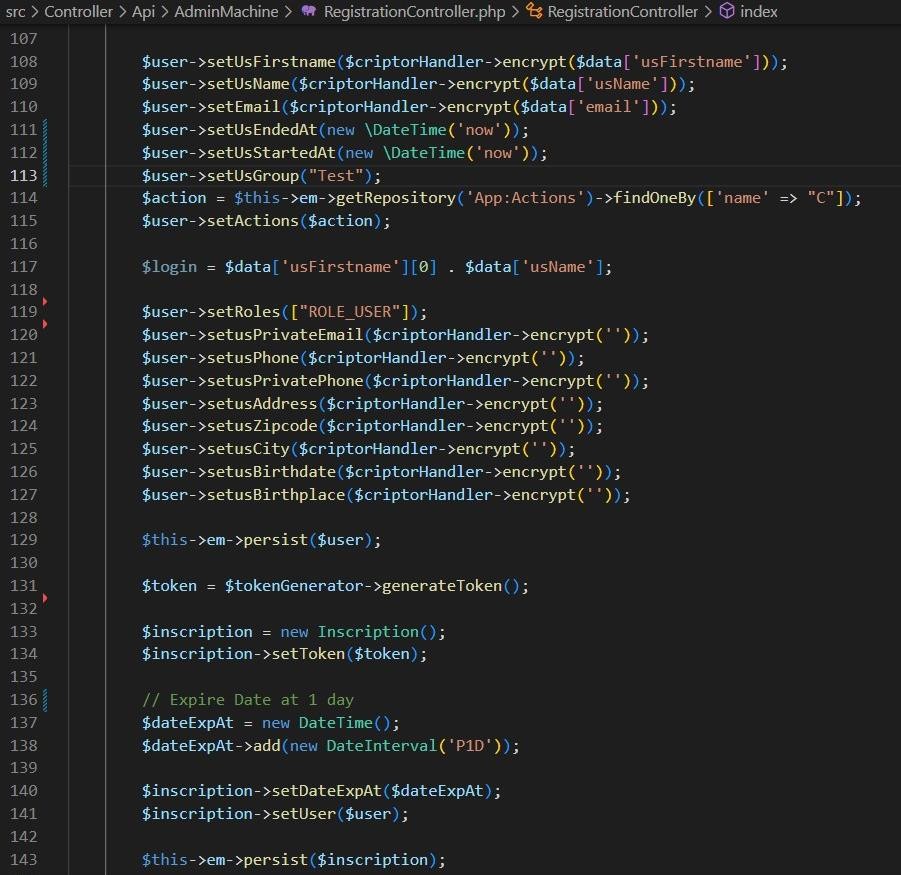
Nous avons placé les controllers propres à l’API selon nos endpoints définis dans le dossier Api.

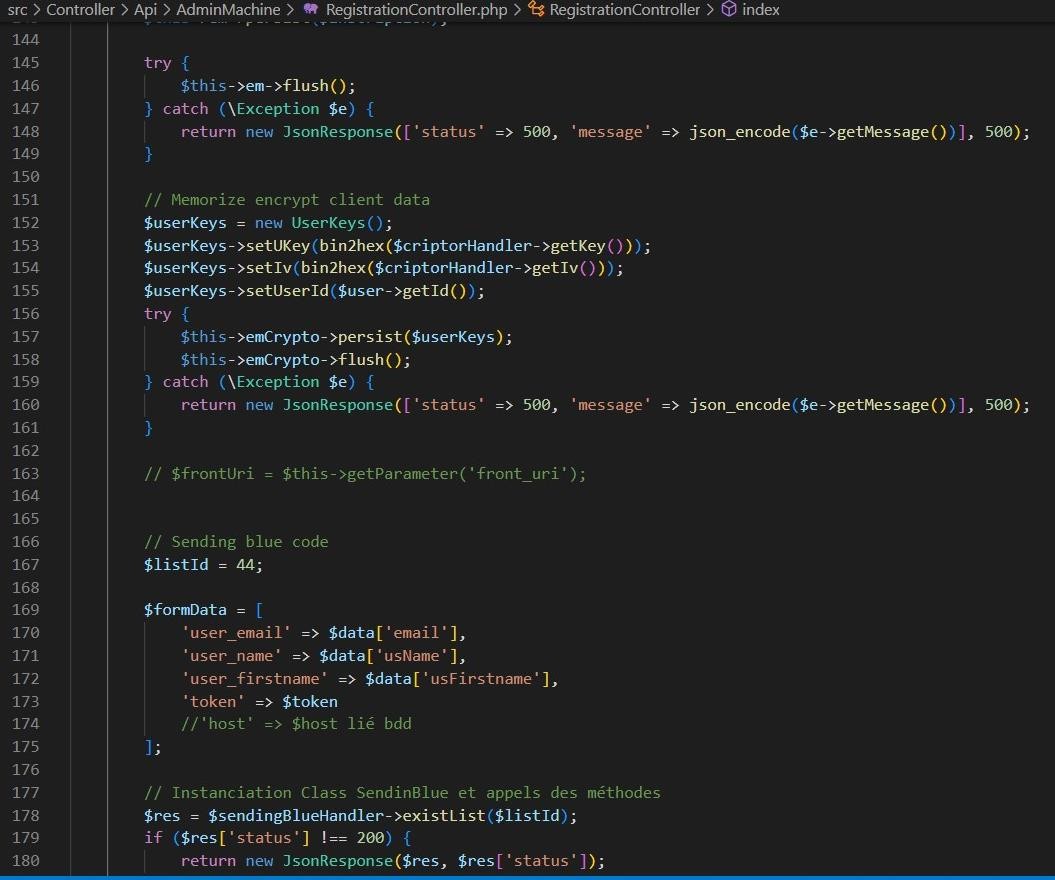
Dans « Controller/Api/Admin\_Machine », nous avons comme exemple de contrôleur :

* **RegistrationController** qui se charge d’ajouter l’utilisateur en base de données à

son inscription.









### Étapes de la méthode « new » pour un créer un user.

1. Récupération des données de formulaire front pour la création d’un utilisateur (nom, prénom, téléphone, adresse mail) encodée en JSON et convertit en variable PHP.
2. Un mot de passe aléatoire est généré et haché lors du register car ne champs ne peut pas être vide.
3. Puis on vérifie l’intégrité des données dans le code l.86 et on valide les données via la fonction **validForm** qui utilise le service de validation de données. En cas d’erreur, on renverra un objet d’erreur avec les erreurs trouvées et de code 400 pour indiquer une mauvaise requête.
4. Un token est généré dans la table inscription pour une durée de 24 heures (l.131)
5. Grâce à **persist** l. 143 on prend une instance d’entité, pour l’ajouter au contexte et gérer cette instance (suivi de l’entité)
6. **Flush** l.146 permet ensuite de synchroniser le contexte de persistance avec la base de données de la table inscription.
7. Ensuite les données des users sont chiffrées (l152 à 161)
8. L’API de SendInBlue est connectée avec une liste de contact (liste numéro 44).
9. Enfin, l.194 on retourne à l’utilisateur la réponse de l’API contenant l’objet créé (user) avec le code 201 indiquant que la requête à réussie la création d’une ressource.

### Développer la partie back-end d’une application web ou web mobile

On retrouve dans Admin\_Machine différents concepts et bonnes pratiques dans la mise en place de la partie back-end.

***Utilisation de la POO (Programmation Orientée Objet)***

La POO permet ici de réutiliser le code dans différents projets. Cela permet également d’avoir un code clair et organisé en identifiant chaque élément présent dans l’application comme un objet ayant son contexte, ses propriétés et des actions qui lui sont propres. De plus, ces objets possèdent leur propre contexte et ne peuvent agir avec d’autres objets que dans des interfaces bien précises et définies par son concepteur.

***Sécurité***

### CORS (Cross-Origin Resource Sharing)

Nous avons utilisé le bundle NelmioCorsBundle qui nous a permis de définir les règles CORS. Ce bundle permet de définir les domaines qui auront accès à notre API REST. Ceux- ci étaient désactivés lors de l’utilisation du serveur en mode développement, mais activés en production avec uniquement acceptation des requêtes provenant du serveur front. Cette opération interdit le chargement à partir d’autres serveurs.

### Token JWT

Le bundle JWT Authentication Bundle nous a offert la possibilité d’utiliser les JSON Web Tokens afin de protéger les ressources de notre API REST. De fait, lorsque l’utilisateur s’authentifie correctement, un token lui est retourné. Ce token est ensuite transmis côté front au serveur afin de permettre à l’utilisateur de lire ou d’écrire les ressources protégées à chaque requête.

### Variable d’environnement

L’utilisation d’une variable d’environnement avec le fichier *.env* à la racine du projet permet de définir en un point un accès, ici la base de données MySQL et les informations de connexion. Ce fichier n’est pas public et ce pour des questions de sécurités. Seuls les développeurs participants à l’application ont accès à ces données.

### security.yaml

Au sein du fichier **config/packages/security.yaml** on décrit les règles d’authentification et d’autorisation pour Admin\_Machine. On y définit l’authentification requise pour les routes présentent dans notre dossier **Api** contenant les endpoints de l’application comme vus précédemment, hormis la vérification à la connexion de l’utilisateur, car celui-ci ne dispose pas encore de token à ce moment-là. C’est aussi là que l’on définit pour notre entité User l’algorithme de hachage de mot de passe utilisé.

### Rôles

On définit dans notre application les rôles que nous utiliserons, leurs hiérarchies et les routes qui leurs sont réservées dans le fichier **config/packages/security.yaml.**

Le rôle utilisateur nommé **ROLE\_USER** est toujours attribué à tout utilisateur inscrit. Ce rôle permet d’utilisation des endpoints définies dans l ’application.

Des rôles ont été défini mais la gestion de l’autorisation des utilisateurs sera faite ultérieurement car le cahier des charges n’a pas été clairement défini sur ce sujet.

# Présentation du jeu d’essai

Dans cette partie, je vais vous présenter le cheminement complet d’une création d’un utilisateur, des captures d’écrans pour illustrer et le code pour l’expliquer.

L’ajout d’un utilisateur passe par la création d’un User et c’est donc le

RegistrationController qui gère cela, et c’est la table User qui est concernée.

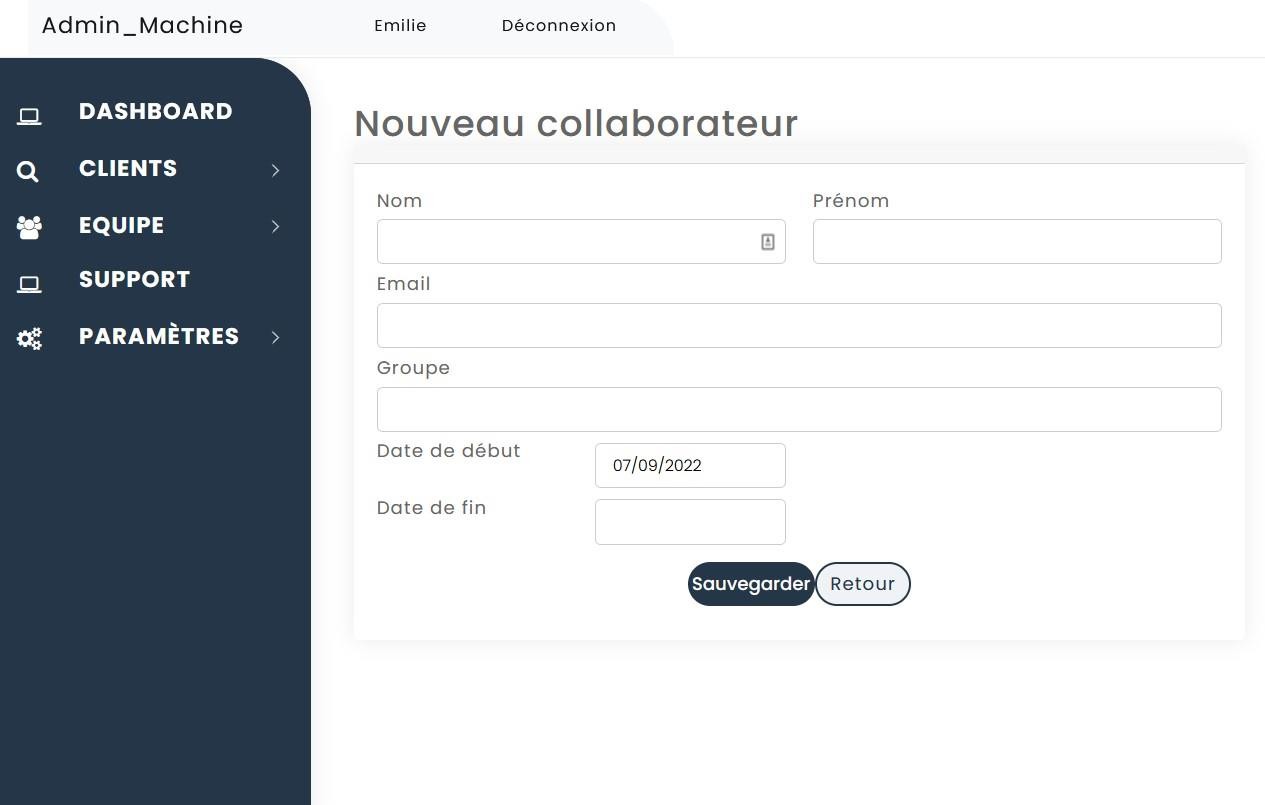
Voici l’interface d’un utilisateur connecté. Concernant l’administrations des membres de l’équipe il y a deux liens dans la sidebar, le premier l’ajout d’un membre et de deuxième l’annuaire de tous les membres.

* 1. L’utilisateur connecté, peut créer un nouveau collaborateur. Il lui suffit dans la

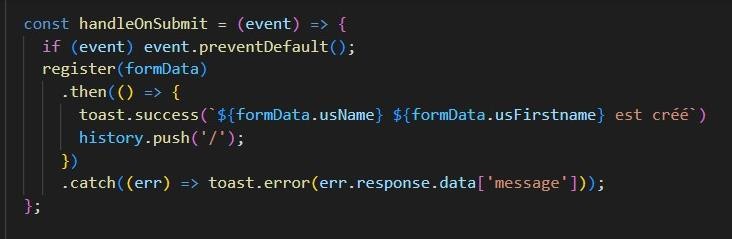
sidebar de cliquer sur ajout d’un membre.



* 1. Il remplit le formulaire d’ajout de l’équipe (voir le régistration controller plus en détails de la page 25 à 27 pour le détail de l’ajout).



Dans le fichier RegistrationForm.jsx nous avons les vérifications de la fonction **validate**, ainsi que le **handleOnSubmit** qui permet d’envoyer ou non le formulaire. Si c’est positif, un toast apparait avec le message définit.

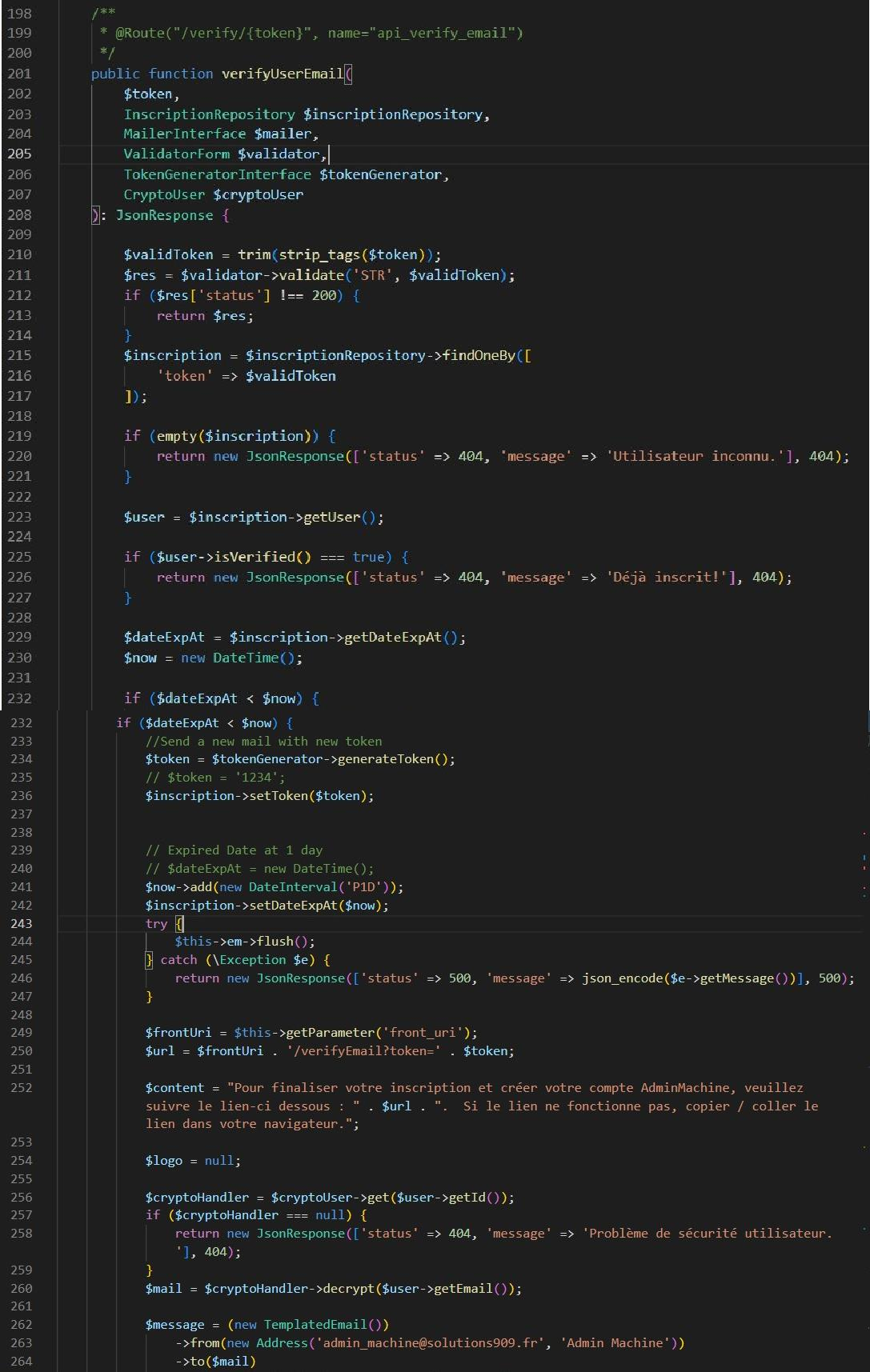


* 1. Un email de création de compte est envoyé par l’API SendInBlue.

Voici le template d’email reçu par l’utilisateur. Ainsi que le code du verifyEmail

prévu dans le Registration Controller qui envoie cet email.



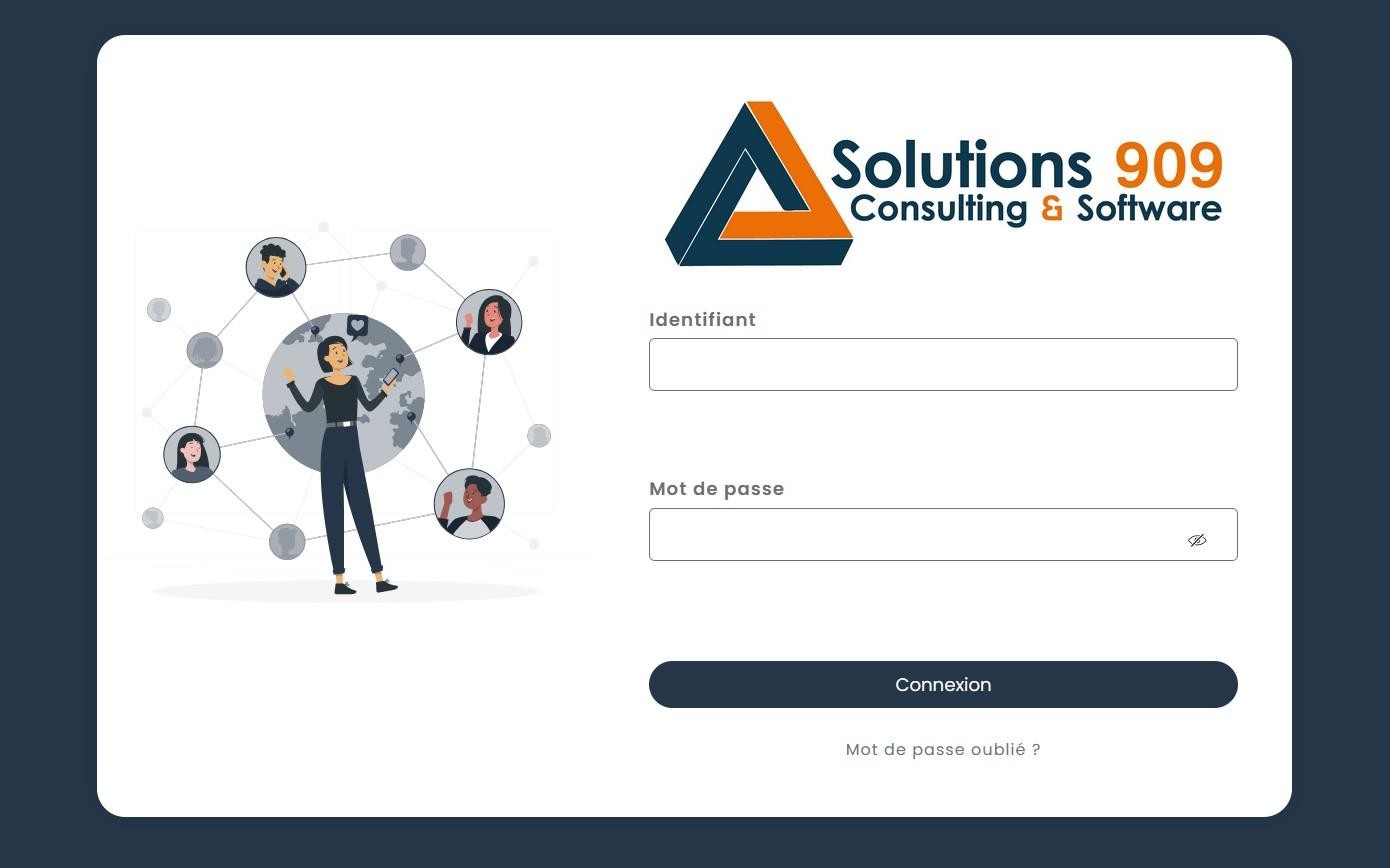




* 1. Le futur collaborateur clique sur l’email et, est redirigé vers le site où il choisit son mot de passe (le login sera communiqué au nouvel utilisateur par l’administrateur qui l’a ajouté).

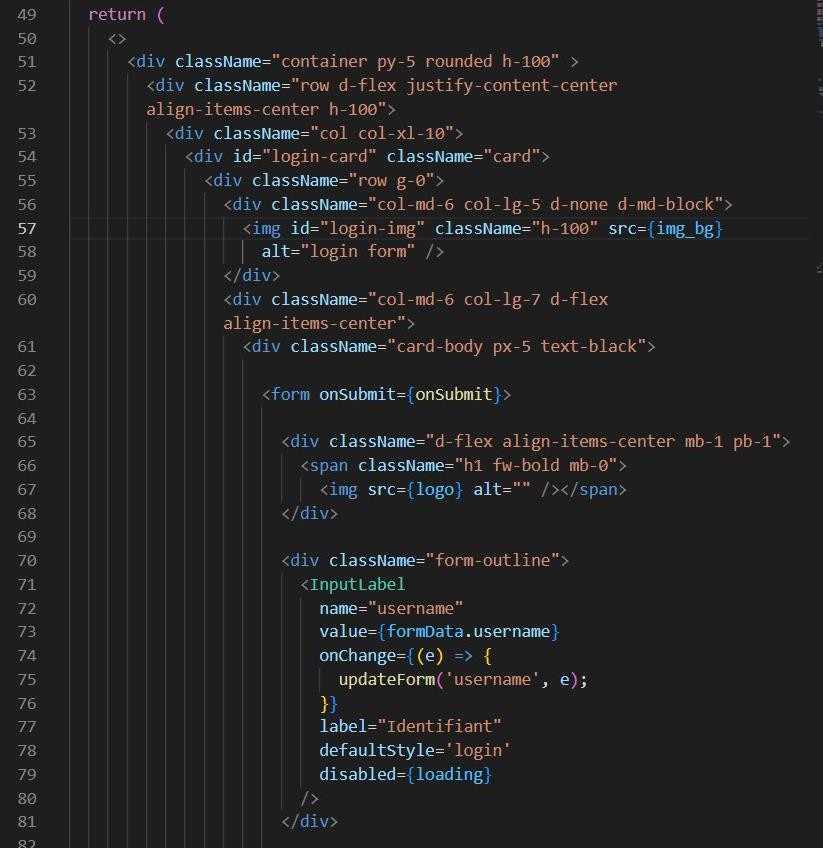


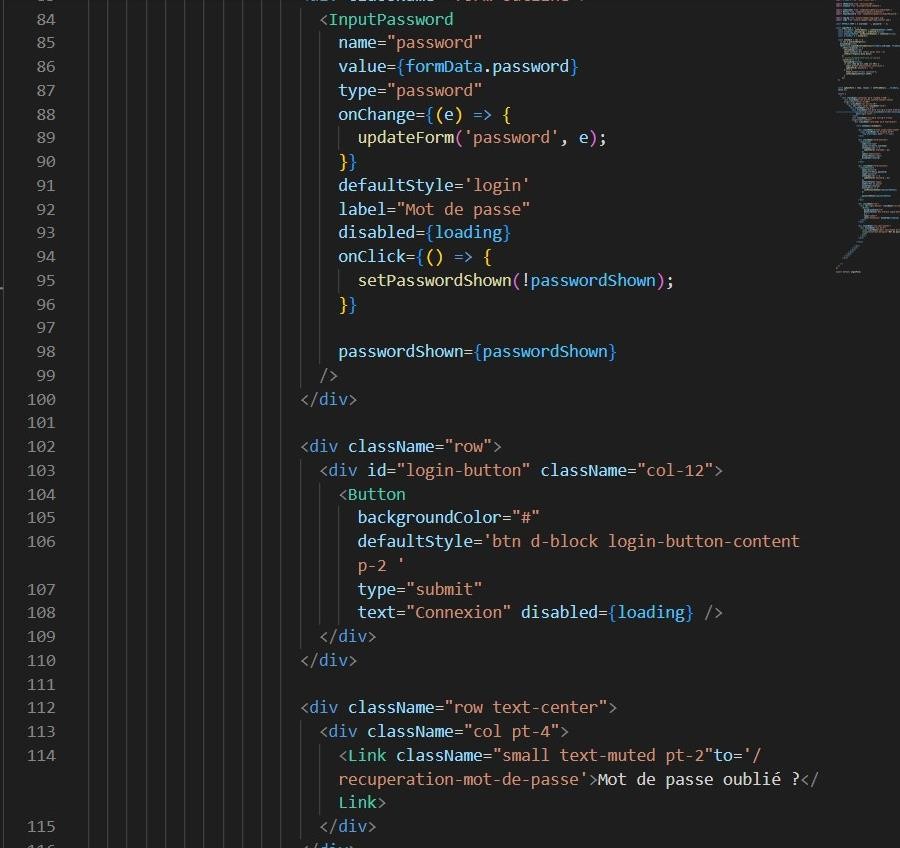
* 1. Ensuite il se connecte avec son login ainsi que son mot de passe choisit.



Visualisation de la page de login







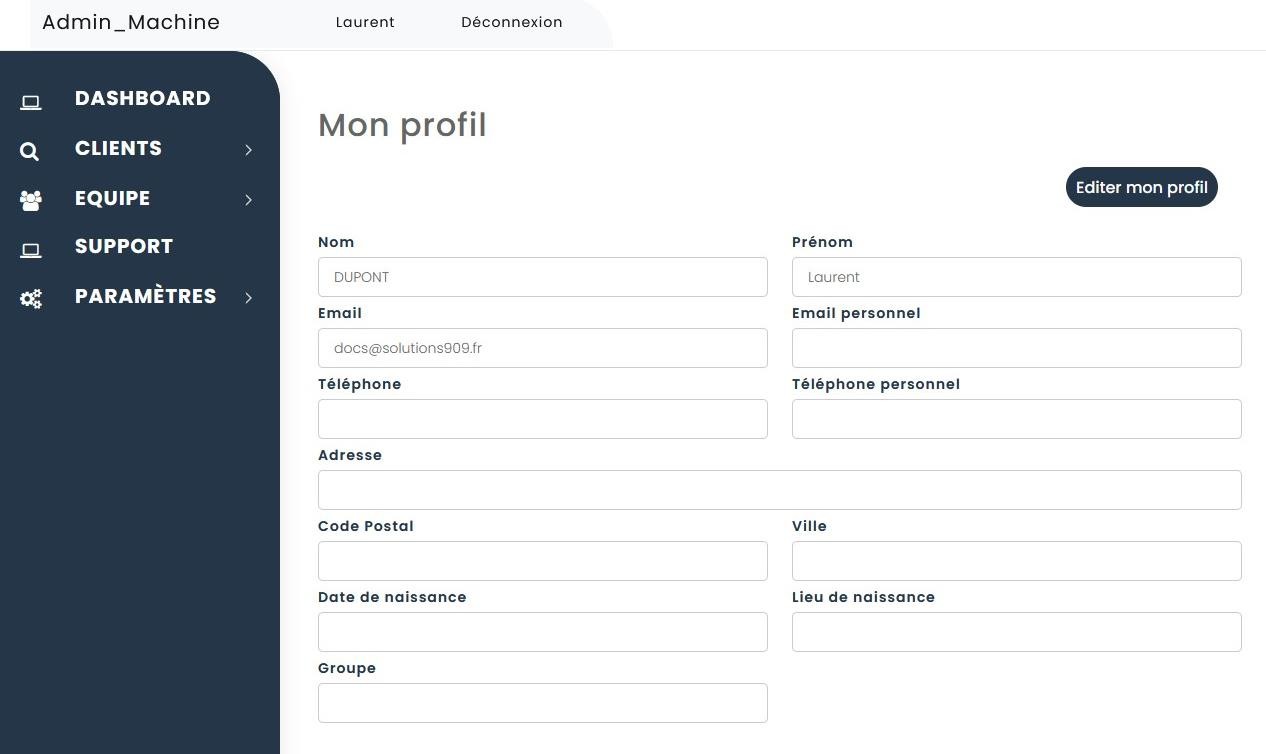
Visualisation du RegistrationForm.jsx, qui implémente la page de login et la fait fonctionner.

L’utilisateur est alors connecté.

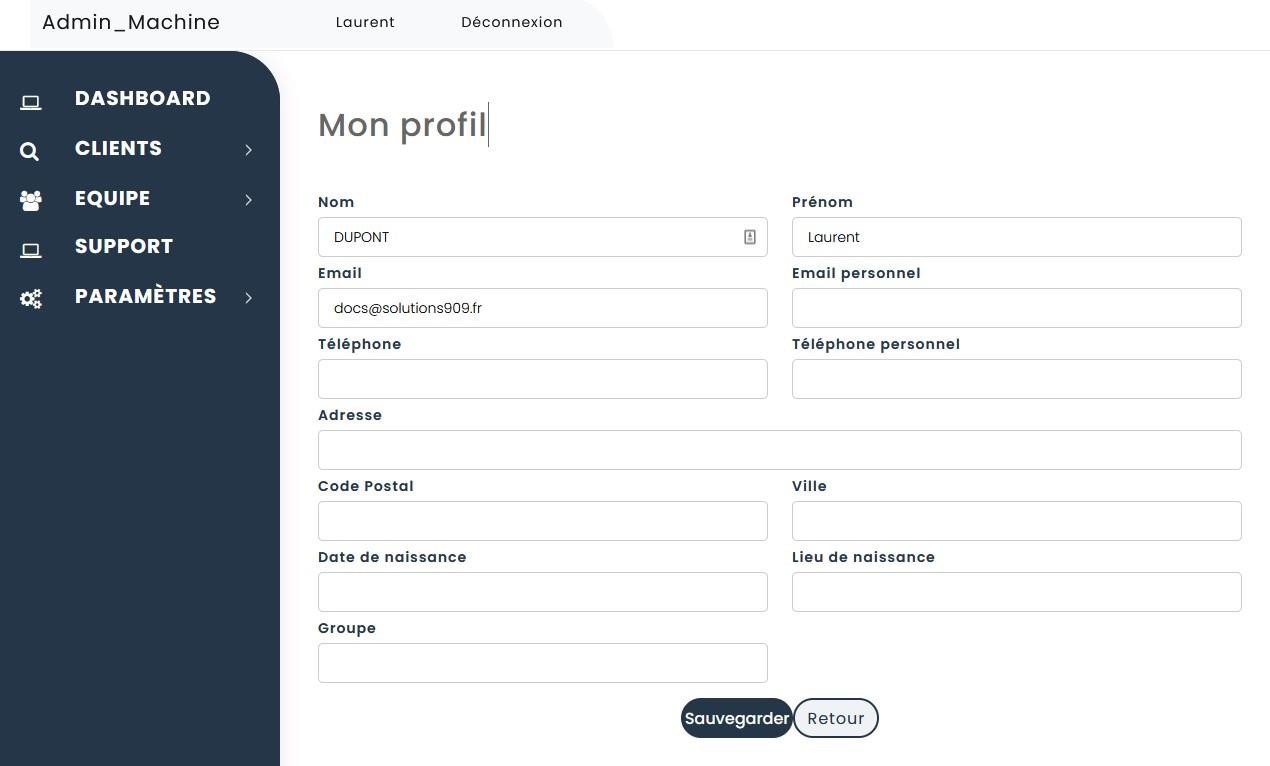


L’utilisateur peut alors finir de compléter son profil en cliquant sur le bouton éditer mon

profil.



Une fois que l’on a cliqué, le bouton éditer disparait et le bouton sauvegarder et retour apparaissent.



# Veille sur les vulnérabilités de sécurité

Solutions 909 est une entreprise spécialisée en cybersécurité et protection des données personnelles. De fait protéger les données des clients et des utilisateurs est un facteur primordial et une attention particulière a été demandée par le dirigeant concernant la sécurité. Protéger les données personnelles et une obligation mais aussi un devoir.

### Validation de l’email et du mot de passe

Un email de vérification est envoyé à l’utilisateur via l’API **SendinBlue**. Une fois la vérification de l’email faite grâce au token, l’utilisateur doit renseigner un mot de passe ainsi que sa confirmation.

Une fois son inscription finalisée, il doit se connecter à l’aide d’un login (fournit par l’administrateur du site) et le mot de passe choisit par le nouvel utilisateur.

Le mot de passe doit avoir un niveau minimum de complexité pour ne pas être facilement deviné. L'utilisateur est prévenu si la complexité n'est pas suffisante selon la validation par des expressions régulières (regex).

Dans la partie front de l'application, pour l'utilisation d'une input via notre composant

**InputLabel** avec le nom *password* utilisé pour le mot de passe utilisateur.

En définissant dans les règles, on vérifie via un pattern Regex que le mot de passe entré dispose bien de au moins :

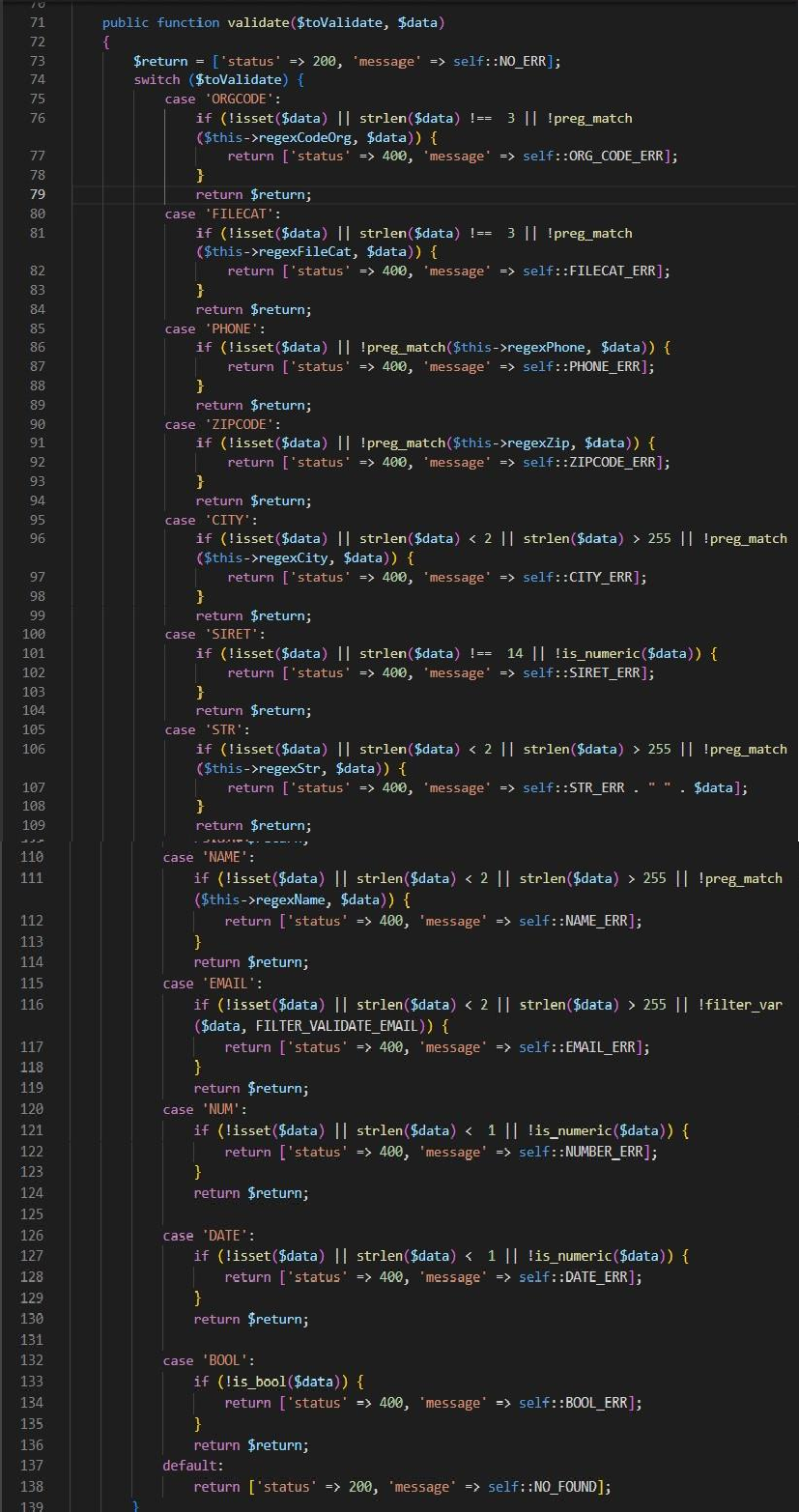
* une lettre majuscule
* une lettre minuscule
* un chiffre
* 10 caractères minimum



Si la condition n'est pas validée, alors une erreur sera affichée sous le champ concerné à la soumission du formulaire et l’utilisateur devra modifier les champs requis.

### Validation en PHP

J'ai également procédé à une validation en PHP avec une **fonction validate** que l’on appelle dans les **validform** des controllers pour vérifier les champs des formulaires dans le back avant de les ajouter en base de données.



# Description d’une situation de travail ayant nécessité une recherche à partir de site anglophone

Lors de mon stage j’ai fait énormément de recherche de documentation notamment sur Boltcms, car c’est une technologie que je n’avais pas étudié (contrairement à Symfony ou j’avais déjà appris les bases).

Lors d’une recherche que j’ai tapé dans mon moteur « How to add tags on boltcms » j’ai trouvé ce site internet anglophone : <https://boltcms.slack.com/> qui est un célèbre lieu de rendez-vous pour tout les développeurs qui travaille avec ce cms .

### Texte original

If you would like to display a list of records with the same taxonomy values, for instance “all pages in the same chapter”, you can do something like this:

{% setcontent mypages = 'pages' where {'chapters': myvalue } %}

<ul>

{% for record in mypages %}

<li><a>{% if record|current %} class="active"{% endif %} href="{{ record|link }}"> {{ record|title }}</a></li>

{% endfor %}

</ul>

If you have a taxonomy with the multiple: true setting you can do something like this:

{% if record|taxonomies.chapters is defined %}

{% for value in record|taxonomies['chapters'] %}

{% setcontent mypages = 'pages' where {'chapters': value.slug } %}

<ul>

{% for record in mypages %}

<li><a class="tag{% if record|current %} active{% endif %}" href="{{ record|link }}"> {{ record|title }}</a></li>

{% endfor %}

</ul>

{% endfor %}

{% endif %}

### 

### Traduction

Si vous souhaitez afficher une liste d'enregistrements avec les mêmes valeurs de taxonomie, par exemple "toutes les pages du même chapitre", vous pouvez faire quelque chose comme ceci :

{% setcontent mypages = 'pages' where {'chapters': myvalue } %}

<ul>

{% for record in mypages %}

<li><a>{% if record|current %} class="active"{% endif %} href="{{ record|link }}"> {{ record|title }}</a></li>

{% endfor %}

</ul>

Si vous avez une taxonomie avec le paramètre multiple : vrai, vous pouvez faire quelque chose comme ceci :

{% if record|taxonomies.chapters is defined %}

{% for value in record|taxonomies['chapters'] %}

{% setcontent mypages = 'pages' where {'chapters': value.slug } %}

<ul>

{% for record in mypages %}

<li><a class="tag{% if record|current %} active{% endif %}" href="{{ record|link }}"> {{ record|title }}</a></li>

{% endfor %}

</ul>

{% endfor %}

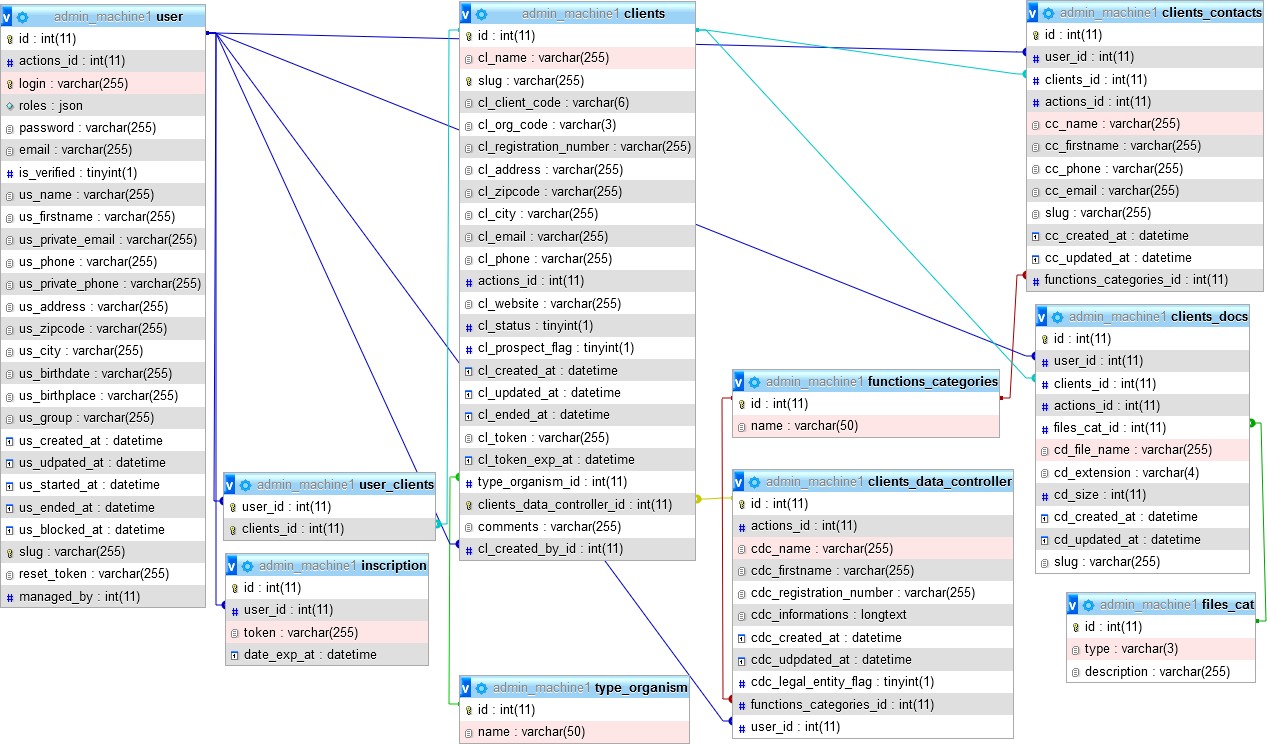
{% endif %}

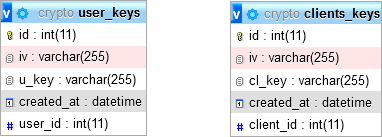
# Conclusion

Pour conclure, Ce projet a été un bon moyen pour moi de progresser dans mon apprentissage en découvrant de nouvelles méthodes de travail (le cms Bolt en particulier) J'ai pu remarquer l'importance d'être polyvalent face aux problématiques rencontrées tout au long de ce projet. Il me reste encore beaucoup de travail à accomplir avant la mise en ligne du site, et beaucoup de chose à apprendre. Ce projet m'aura également conforté dans ce choix professionnel, à la fois intense et passionnant.

# XII. Annexes

### Base de données





### Dictionnaire de données

**actions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id actions |
| name | varchar(4) | Non |  | nom de l'action |
| description | varchar(100) | Non |  | description de l'action |

**clients**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id client |
| cl\_name | varchar(255) | Non |  | nom client |
| slug | varchar(255) | Non |  |  |
| cl\_client\_code | varchar(6) | Non |  | code client |
| cl\_org\_code | varchar(3) | Non |  | code organisme principal |
| cl\_registration\_number | varchar(255) | Non |  | SIRET du client |
| cl\_address | varchar(255) | Non |  | adresse client |
| cl\_zipcode | varchar(255) | Non |  | code postal client |
| cl\_city | varchar(255) | Non |  | ville client |
| cl\_email | varchar(255) | Non |  | email client |
| cl\_phone | varchar(255) | Oui | *NULL* | téléphone client |
| actions\_id | int(11) | Oui | *NULL* | id actions |
| cl\_website | varchar(255) | Oui | *NULL* | url du site client |
| cl\_status | tinyint(1) | Non |  | false-> inactif ou true -> actif |
| cl\_prospect\_flag | tinyint(1) | Non |  | false-> prospect ou true -> client |
| cl\_created\_at | datetime | Non |  | date de création du client |
| cl\_updated\_at | datetime | Oui | *NULL* | date de mise à jour du client |
| cl\_ended\_at | datetime | Oui | *NULL* | date de désactivation du client |
| cl\_token | varchar(255) | Oui | *NULL* |  |
| cl\_token\_exp\_at | datetime | Oui | *NULL* |  |
| type\_organism\_id | int(11) | Oui | *NULL* | id type d'organisme |
| clients\_data\_controller\_id | int(11) | Oui | *NULL* | id data controller |
| comments | varchar(255) | Oui | *NULL* | annotations client |
| cl\_created\_by\_id | int(11) | Oui | *NULL* | id utilisateur |

**clients\_contacts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id contact |
| user\_id | int(11) | Oui | *NULL* | id utilisateur |
| clients\_id | int(11) | Oui | *NULL* | id client |
| actions\_id | int(11) | Oui | *NULL* | id actions |
| cc\_name | varchar(255) | Non |  | nom contact |
| cc\_firstname | varchar(255) | Non |  | prénom contact |
| cc\_phone | varchar(255) | Oui | *NULL* | téléphone contact |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cc\_email | varchar(255) | Oui | *NULL* | email contact |
| slug | varchar(255) | Non |  |  |
| cc\_created\_at | datetime | Non |  | date de création contact |
| cc\_updated\_at | datetime | Oui | *NULL* | date de mise à jour contact |
| functions\_categories\_id | int(11) | Oui | *NULL* | id catégorie de fonction client |

**clients\_data\_controller**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id data controller |
| actions\_id | int(11) | Oui | *NULL* | id actions |
| cdc\_name | varchar(255) | Oui | *NULL* | nom data controller |
| cdc\_firstname | varchar(255) | Oui | *NULL* | prénom data controller |
| cdc\_registration\_number | varchar(255) | Oui | *NULL* | SIRET data controller |
| cdc\_informations | longtext | Oui | *NULL* | informations data controller |
| cdc\_created\_at | datetime | Non |  | date de création data controller |
| cdc\_udpdated\_at | datetime | Oui | *NULL* | date de mise à jour data controller |
| cdc\_legal\_entity\_flag | tinyint(1) | Non |  | false-> personne morale ou true -> personne physique |
| functions\_categories\_id | int(11) | Oui | *NULL* | id catégorie de fonction client |
| user\_id | int(11) | Oui | *NULL* | id utilisateur |

**clients\_docs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id document |
| user\_id | int(11) | Non |  | id utilisateur |
| clients\_id | int(11) | Non |  | id client |
| actions\_id | int(11) | Non |  | id actions |
| files\_cat\_id | int(11) | Non |  | id catégorie de document |
| cd\_file\_name | varchar(255) | Non |  | nom du document enregistré |
| cd\_extension | varchar(4) | Non |  | extension du document |
| cd\_size | int(11) | Non |  | taille du document |
| cd\_created\_at | datetime | Non |  | date de création du document |
| cd\_updated\_at | datetime | Oui | *NULL* | date de mise à jour du document |
| slug | varchar(255) | Non |  |  |

**files\_cat**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id catégorie de document |
| type | varchar(3) | Non |  | type catégorie de document |
| description | varchar(255) | Non |  | description catégorie de document |

**functions\_categories**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id catégorie de fonction client |
| name | varchar(50) | Non |  | nom catégorie de fonction client |

**inscription**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  |  |
| user\_id | int(11) | Non |  | id utilisateur |
| token | varchar(255) | Non |  |  |
| date\_exp\_at | datetime | Non |  |  |

**log**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id log |
| user\_id | int(11) | Oui | *NULL* | id utilisateur |
| message | varchar(255) | Non |  | message log |
| context | longtext | Non |  | contexte log(DC2Type:array) |
| level | smallint(6) | Non |  | niveau log |
| level\_name | varchar(50) | Non |  | nom du niveau log |
| extra | longtext | Oui | *NULL* | extra log(DC2Type:array) |
| created\_at | datetime | Non |  | date de création log |

**refresh\_tokens**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  |  |
| refresh\_token | varchar(128) | Non |  |  |
| username | varchar(255) | Non |  |  |
| valid | datetime | Non |  |  |

**reset\_password\_request**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  |  |
| user\_id | int(11) | Non |  | id utilisateur |
| selector | varchar(20) | Non |  |  |
| hashed\_token | varchar(100) | Non |  |  |
| requested\_at | datetime | Non |  | (DC2Type:datetime\_immutable) |
| expires\_at | datetime | Non |  | (DC2Type:datetime\_immutable) |

**type\_organism**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id type d'organisme |
| name | varchar(50) | Non |  | nom type d'organisme |

**user**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| id *(Primaire)* | int(11) | Non |  | id utilisateur |
| actions\_id | int(11) | Non |  | id actions |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| login | varchar(255) | Non |  | identifiant utilisateur |
| roles | longtext | Non |  | roles utilisateur |
| password | varchar(255) | Non |  | mot de passe utilisateur |
| email | varchar(255) | Non |  | email utilisateur |
| is\_verified | tinyint(1) | Non |  |  |
| us\_name | varchar(255) | Non |  | nom utilisateur |
| us\_firstname | varchar(255) | Non |  | prénom utilisateur |
| us\_private\_email | varchar(255) | Oui | *NULL* | email privé utilisateur |
| us\_phone | varchar(255) | Non |  | téléphone utilisateur |
| us\_private\_phone | varchar(255) | Oui | *NULL* | téléphone privé utilisateur |
| us\_address | varchar(255) | Non |  | adresse utilisateur |
| us\_zipcode | varchar(255) | Non |  | code postal utilisateur |
| us\_city | varchar(255) | Non |  | ville utilisateur |
| us\_birthdate | varchar(255) | Oui | *NULL* | date de naissance utilisateur |
| us\_birthplace | varchar(255) | Oui | *NULL* | lieu de naissance utilisateur |
| us\_group | varchar(255) | Oui | *NULL* | groupe utilisateur |
| us\_created\_at | datetime | Non |  | date de création utilisateur |
| us\_udpated\_at | datetime | Oui | *NULL* | date de mise à jour utilisateur |
| us\_started\_at | datetime | Oui | *NULL* | date de début utilisateur |
| us\_ended\_at | datetime | Oui | *NULL* | date de fin utilisateur |
| us\_blocked\_at | datetime | Oui | *NULL* | date de blocage utilisateur |
| slug | varchar(255) | Non |  |  |
| reset\_token | varchar(255) | Oui | *NULL* |  |
| managed\_by | int(11) | Oui | *NULL* | géré par utilisateur |

**user\_clients**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colonne** | **Type** | **Null** | **Valeur par défaut** | **Commentaires** |
| user\_id *(Primaire)* | int(11) | Non |  | id utilisateur |
| clients\_id *(Primaire)* | int(11) | Non |  | id client |

### Routes (endpoints)

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **HTTP**  **Method** | **Controller** | **URL** |
| api\_auth\_login | POST | AuthController | /api/auth/login |
| api\_forgotten\_password | POST | AuthController | /api/auth/forgotten\_passw ord |
| api\_reset\_password | POST | AuthController | /api/auth/reset\_password/  {token} |
| gesdinet\_jwt\_refresh\_token | POST | AuthController | /api/auth/refresh\_token |
| user\_register | POST | RegistrationController | /api/register |
| api\_verify\_email | ANY | RegistrationController | /api/verify/{token} |
| api\_confirm\_creds | ANY | RegistrationController | /api/confirm |
| api\_user\_edit | PATCH | UserController | /api/user/{id}/edit |
| api\_user\_delete\_fake | PATCH | UserController | /api/user/del/{id} |
| api\_user\_index | GET | UserController | /api/users |

|  |  |  |  |
| --- | --- | --- | --- |
| api\_user\_profile\_edit | PATCH | UserController | /api/user/profil/edit |
| api\_clients\_show | GET | ClientsController | /api/clients/{id} |
| api\_clients\_new | GET|POST | ClientsController | /api/clients/new |
| api\_clients\_edit | PATCH | ClientsController | /api/clients/{id}/edit |
| api\_clients\_comment | PATCH | ClientsController | /api/clients/{id}/comment |
| api\_clients\_manager | PATCH | ClientsController | /api/clients/{id}/manager |
| api\_clients\_delete\_fake | PATCH | ClientsController | /api/clients/del/{id} |
| api\_clients\_remove | DELETE | ClientsController | /api/clients/remove/{id} |
| api\_clients\_dc\_new | GET|POST | ClientsDataControllerController | /api/clients/{id}/dc/new |
| api\_clients\_dc\_edit | PATCH | ClientsDataControllerController | /api/clients/{id}/dc/edit |
| api\_clients\_docs\_index | GET | ClientsDocsController | /api/clients/{id}/docs |
| api\_clients\_docs\_add | POST | ClientsDocsController | /api/clients/{id}/docs/add |
| api\_clients\_docs\_delete\_fake | PATCH | ClientsDocsController | /api/docs/del/{id} |
| api\_clients\_docs\_remove | DELETE | ClientsDocsController | /api/docs/remove/{id} |
| api\_clients\_contacts\_new | GET|POST | ClientsContactsController | /api/clients/{id}/contacts/n  ew |
| api\_clients\_contacts\_edit | PATCH | ClientsContactsController | /api/clients/{id}/contacts/e  dit |
| api\_files\_cat\_index | GET | FilesCatController | /api/filescat/ |
| api\_files\_cat\_new | GET|POST | FilesCatController | /api/filescat/new |
| api\_files\_cat\_edit | PATCH | FilesCatController | /api/filescat/{id}/edit |
| api\_files\_cat\_delete | DELETE | FilesCatController | /api/filescat/remove/{id} |
| api\_functions\_categories\_ind  ex | GET | FunctionsCategoriesController | /api/functions-categories/ |
| api\_functions\_categories\_ne  w | GET|POST | FunctionsCategoriesController | /api/type-organisms/new |
| api\_functions\_categories\_edit | PATCH | FunctionsCategoriesController | /api/type-  organisms/{id}/edit |
| api\_functions\_categories\_del  ete | DELETE | FunctionsCategoriesController | /api/type-  organisms/remove/{id} |
| api\_typeorganism\_index | GET | TypeOrganismController | /api/type-organisms/ |
| api\_log\_index | GET | LogController | /api/log/ |