

2021 - 2022



Rapport d'alternance

Hourcade Mélina

Liste des compétences

AT1

- ✓ Maquetter une application
- ✓ Développer une interface utilisateur de type desktop
- ✓ Développer des composants d'accès aux données
- ✓ Développer la partie front-end d'une interface utilisateur web
- ✓ Développer la partie back-end d'une interface utilisateur web

AT2

- ✓ Concevoir une base de données
- ✓ Mettre en place une base de données
- ✓ Développer des composants dans le langage d'une base de données

AT3

- ✓ Collaborer à la gestion d'un projet informatique et à l'organisation de l'environnement de développement
- ✓ Concevoir une application
- ✓ Développer des composants métiers
- ✓ Construire une application organisée en couche
- ✓ Développer une application mobile
- ✓ Préparer et exécuter les plans de tests d'une application
- ✓ Préparer et exécuter le déploiement d'une application

Table des matières

Liste des compétences	2
Résumé en Anglais	4
Présentation de l'entreprise.....	5
Projet SpiesImmo	8
Cahier des charges.....	8
Conception générale	10
Données des utilisateurs	10
Exigences contraintes	11
Gestion de projet.....	12
Architecture générale.....	15
Architecture du client :	15
Conception BDD	16
Conception Back-end	20
Api Plateforme Symfony.....	20
Spécifications fonctionnelles.....	22
Réalisation Back-end	24
La sérialisation	24
Les opérations API Plateforme :	25
Connexion avec JwtToken	27
PostMan :	29
Conception front-end	31
Choix des technos.....	31
AVANTAGES :	31
INCONVÉNIENTS :	31
redux :	32
Arborescence :	32
Pourquoi expo ?	33
ReactNative IOS et Android.....	34
Réalisation Front-end:	34
Formulaire de connexion	34
Get des contact avec axios	37
Test de l'application	41
Veille	41
Veille Sécurité.....	41

Résumé en Anglais

I will introduce you to my company, I did my internship for more than two years at hackers-corporation. Hackers-corporation

is a company specialized in the development and implementation of solutions for other companies.

We can do application development, but also task automation for some, which takes time.

My position at hackers-corporation to evolve at the same time as the company, I started by doing development

and now I manage the teams and communication with management.

After discussing with my hierarchie, I would continue with hackers-corporation after the end of my studies.

Présentation de l'entreprise

Objectif de l'entreprise :

1 objectif principal et 2 importants :

- Principal : faire évoluer, dans tous les domaines, les personnes du secteur informatique qui se sentent bloqués ou limités en matière de performance.
- Important 1 : être présent dans son marché et apporter un impact, une différence, à l'international.
- Important 2 : faire évoluer ses collaborateurs en terme d'expertise et de développement personnel.

Vision de l'entreprise :

HACKERs Management œuvre pour apporter davantage d'action dans les entreprises de l'informatique, pour oser faire différemment, innover et apporter de l'autonomie pour faire ressortir le potentiel et la performance des hommes et des femmes en entreprise.

Valeurs de l'entreprise :

- Audace et Transformation
- Pragmatisme et Innovation
- Expérimentation et Hacking
- Confiance et Engagement
- Expérience utilisateur

METHODE DE L'ENTREPRISE :

Afin d'améliorer la performance opérationnelle en optimisant les processus des équipes de travail, HACKERs Management va identifier les leviers de transformation c'est-à-dire les points-clés à travailler dans une entreprise (la production, le management...) et résoudre le problème à sa source pour une augmentation de la performance rapide et durable

Mon évolution :

2020/2021

Developpeur



Developpement cakephp/Laravel



Création Wordpress



Tenue du Serveur

2021/2020

Controleur de Gestion/Performance



Suivis et planification des projets



Contrôles et comptes rendus aux equipes



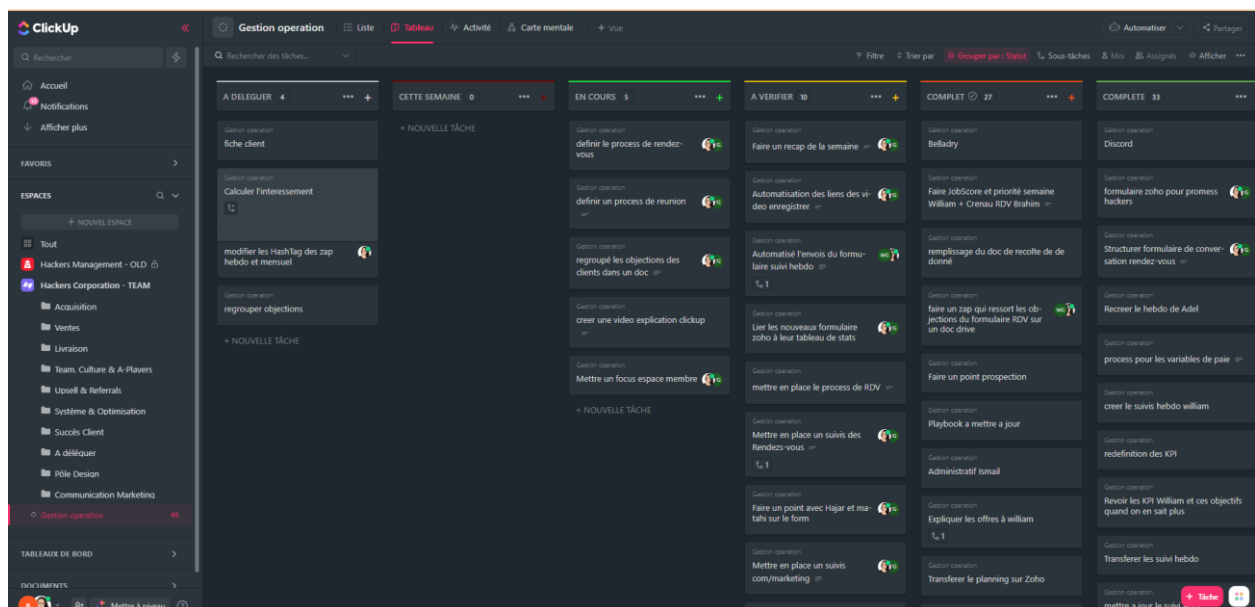
Automatisation et simplification de process



Amélioration des outils quotidiens

Mon quotidien :

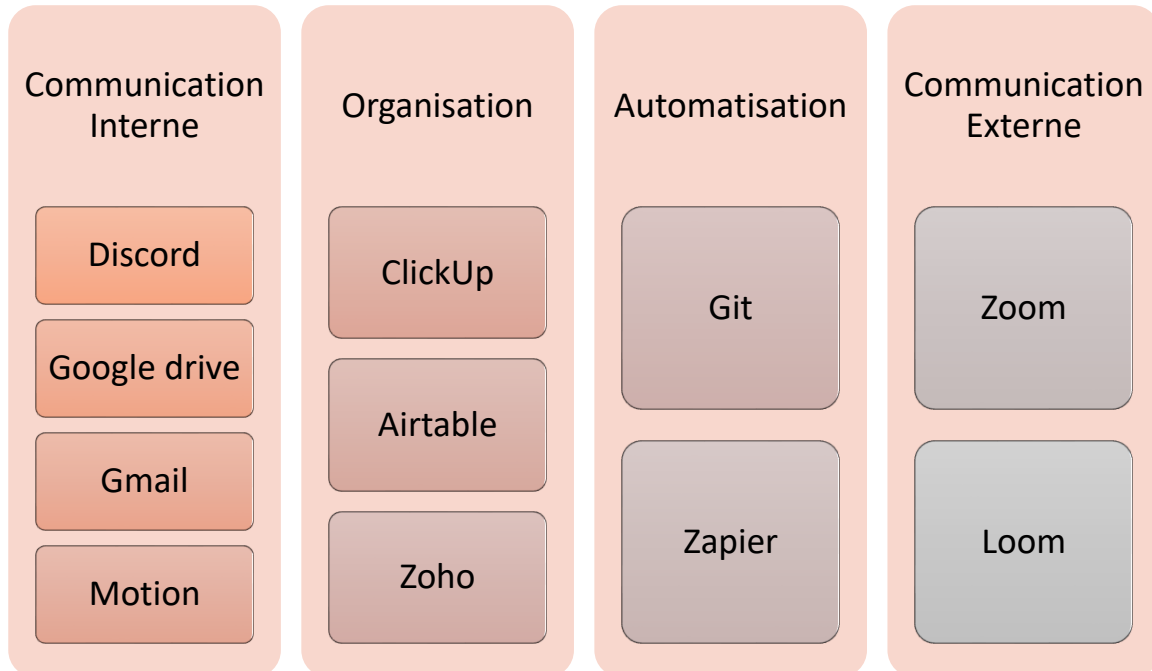
Methode Kanban/ PDCA du Lean Management –



La **méthode PDCA** est une approche itérative pour l'amélioration continue des produits, des individus et des services. Elle est devenue une partie intégrante de ce qui est convenu aujourd'hui d'appeler

le **lean** management. La **méthode** comprend le test de solutions, l'analyse des résultats et l'amélioration du processus.

Les outils :



Projet SpiesImmo

Cahier des charges

Contexte

C'est au contact de plusieurs agents immobiliers que nous avons pris conscience d'un réel problème pour certains agents. Au cours de ces entretiens (nous avons essayé de diversifier au maximum nos agents pour avoir un panel utilisateur le plus large possible) nous nous sommes rendu compte de plusieurs problèmes récurrents pour les agents.

Après avoir relevé les différents soucis que pouvaient rencontrer ces agents nous avons eu une réunion pour prévoir une première version qui a été validée avec l'équipe.

Cette version concerne uniquement le fait qu'un agent puisse gérer ses clients et les documents appartenant à ce client.

Plusieurs versions sont prévues par la suite, mais pour l'instant nous nous focaliserons sur le MVP.

Analyse du besoin

Un agent immobilier doit gérer ses clients et leurs documents, ils doivent suivre leurs contrats, mais ont des difficultés à rester à jour.

Ils veulent gagner du temps lors de la location ou d'une vente d'un bien immobilier.

Certains clients mettent du temps à rendre la totalité des documents, il faut toujours les relancer.

Un Agent Immobilier :

- Doit conserver tous les documents d'une vente immobilière ou location
- Doit effectuer une vérification quotidienne de ces mails pour récupérer les documents d'un client
- Doit faire des rappels fréquemment aux clients qui oublient d'envoyer des documents, ou qui les ont envoyés mais mal remplis
- Perte de temps dans le regroupement des documents adéquats à une vente ou une location

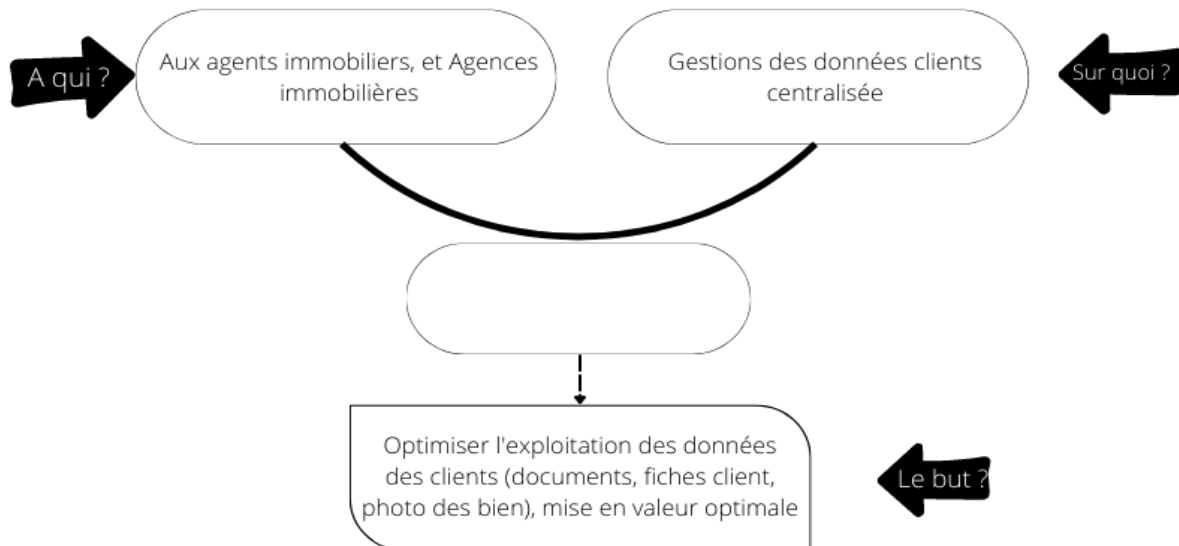
Comment répondre à ce besoin ?

Nous avons élaboré une solution, en couplant la demande des clients, les possibilités techniques, la capacité des développeurs et le temps imparti.

Nous proposons un MVP, une application mobile qui permettra aux Agents de pouvoir créer et gérer leur client, ainsi que d'ajouter les documents correspondant au client. L'application Spies Immo, propose une fluidité et une rapidité d'accès aux informations des clients pour les agents. D'un coup d'œil un agent peut voir si un client a des documents manquants ou non.

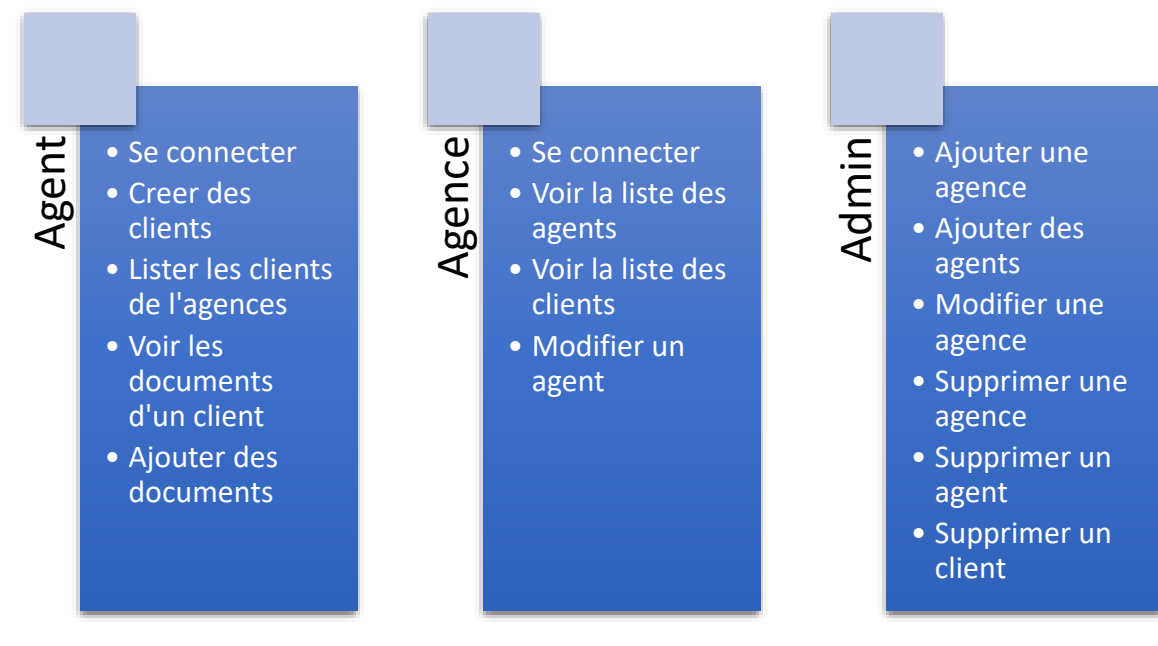
Une V2 dans laquelle un agent peut récupérer tous les documents d'un client dans un seul PDF et l'envoyer par mail à une tiers personne.

Une V3 qui permettrait aux agents de s'associer à une agence, dans le but de partager des informations comme : Un agenda de RDV ou de vacances.



Conception générale

Exigences fonctionnelles



**Agence ne fait pas partie du MVP pour l'instant*

Données des utilisateurs



Dans le cadre de la récolte des données dans l'application, Spies Immo comportera des données sensibles car un agent immobilier devra garder une copie de la pièce d'identité d'un client :

- Sous l'empire du RGPD (entrant en vigueur au 25 mai 2018), les données pouvant permettre une usurpation d'identité (**ce qui est le cas des copies de pièces d'identité**) sont **expressément considérées comme des données sensibles, dont le traitement est en principe interdit**. C'était déjà la position de la CNIL.
- Sous l'empire de la loi de 78 comme sous celui du RGPD, il est donc possible de recueillir la date de naissance des clients, mais pas de conserver de copie de leur titre d'identité.

Que faire dans ce cas-là ?

- contact CNIL

Les données de nos utilisateurs seront conservées 6 mois après leur désinscription.

Pour que le consentement soit valide selon le RGPD, il doit être :

- Donné librement • Spécifique • Informé • Sans équivoque

Exigences contraintes

EC1	Restreindre les actions utilisateurs suivant le niveau d'habilitation
EC2	L'application doit comprendre une API en Symfony
EC3	L'API doit utiliser API Plateforme
EC4	La version mobile doit être codée avec React Native
EC5	La version Web doit être codée avec Angular + Angular Matériel
EC6	La version Web ne doit être accessible qu'aux admin
EC7	Les tests de la version web doivent être effectués avec Cypress
EC8	Les tests en react native doivent être effectués avec React native testing library & Jest
EC9	Les tests de L'API doivent être effectués avec PostMan
EC10	Les requêtes de L'API doivent être répertoriés dans un fichier Swagger
EC11	Respecter la réglementation RGPD pour les informations des utilisateurs
EC12	Assurer l'intégrité des données
EC13	L'utilisateur doit être notifié des erreurs d'utilisation.
EC14	L'ihm doit être intuitive
EC15	Le partage des clients entre les agents d'une agence doit être réglementé <small>*En Réflexion niveau RGPD et ne concerne pas le MVP</small>
EC16	Une fois désinscrit les données de l'agent ou l'agence seront conservées 6 mois
EC17	<i>En cas de non-connexion pendant 1 an, le compte et les données de l'agence ou l'agent sont automatiquement supprimés</i>

Gestion de projet

Equipe :



Outils :



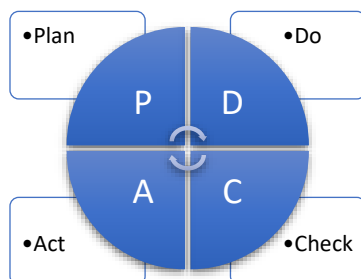
Nous avons décidé d'utiliser beaucoup discord pour la communication interne, car c'est un outil très permissif, auquel on peut ajouter beaucoup de plugin comme zapier, ou google agenda qui permet une gestion du temps optimum.

Le drive de google, nous a servi à regrouper toute la documentation créée tout au cours de l'année.

Pour le développement nous avons choisi de travailler avec Visual code studio, et git car ce sont des outils que nous utilisons en entreprise, et que nous maitrisons déjà avec quelque notion avancée.

Organisation :

Méthode Lean Management : PDCA



- Pour les semaines de cours, des comptes rendus écrits du travail effectué

- Des Réunions bimensuelles sur l'avancée du projet, les objectifs à atteindre

On a commencé par les maquettes, car on voulait déjà voir où l'application aurait pu nous mener.

Grâce à ces maquettes, on a pu commencer à construire un MCD.

Pendant cette partie du développement les réunions ont été beaucoup plus fréquentes. Il y avait pas mal de questions techniques à prendre en compte concernant les choix pour l'application. Cette méthodologie nous a permis une marge de manœuvre assez large concernant les décisions.

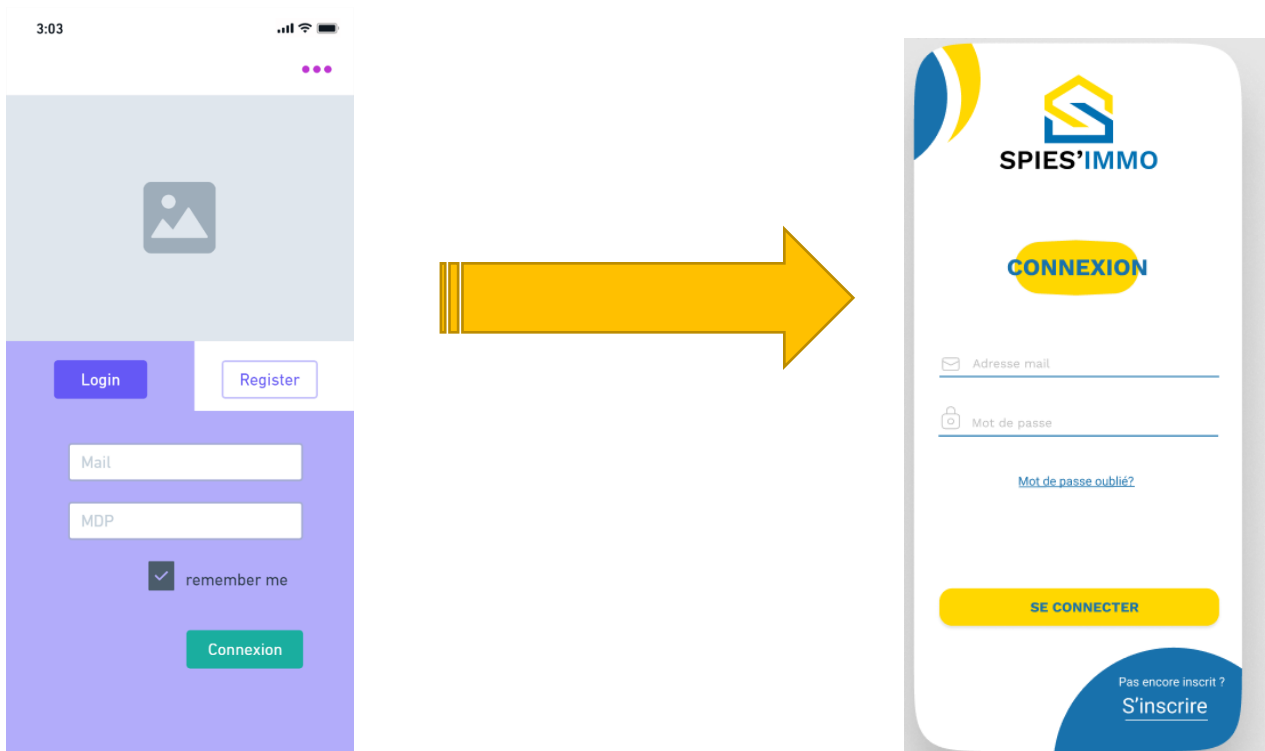
Les bonnes pratiques :

Pour permettre une organisation le tenu des clickUp était très importante, et une bonne utilisation de l'outils aussi. Pour cela, nous devions Tous les matins tenus à jour des taches mais aussi assigner les bonnes personnes à ses taches.

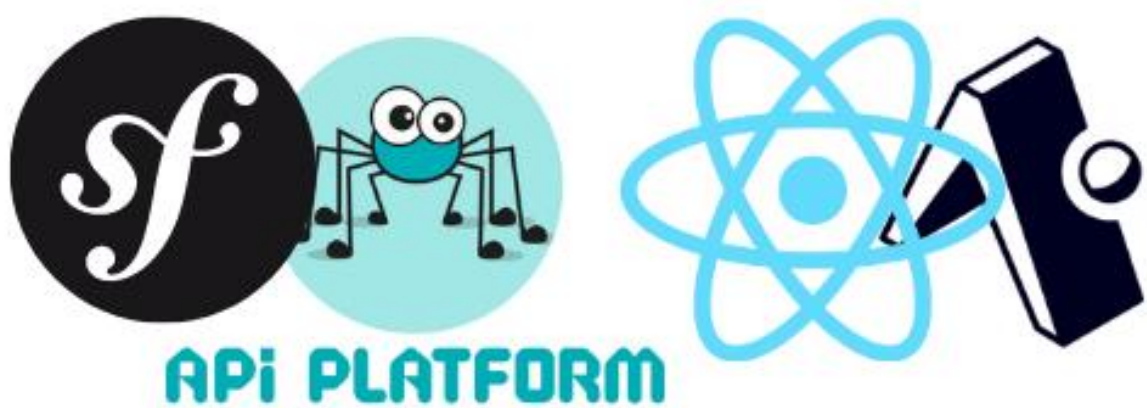
En Bonne pratique nous avons instaurer et agrémenter discord de plusieurs salon, textuel, vocaux, qui on évoluer au fil de l'application, ce qui nous à permis de nous y retrouver facilement.

Utiliser le salon discord approprié aux situations (ex : tips, back-end, front-end, css, autres etc ..)

Wireframe - Maquette :



Les technologies

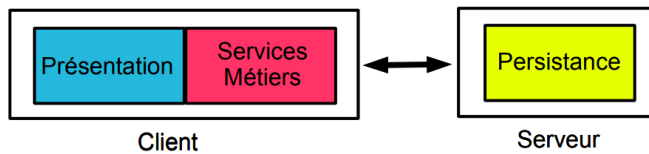


Architecture générale

Architecture 2 – tiers

- Client : présentation, interface utilisateur
- Serveur : partie persistance, gestion physique des données
- Les services métier / la partie applicative sont
 - Entièrement coté client, intégrés avec la présentation
 - La partie serveur ne gère que les données

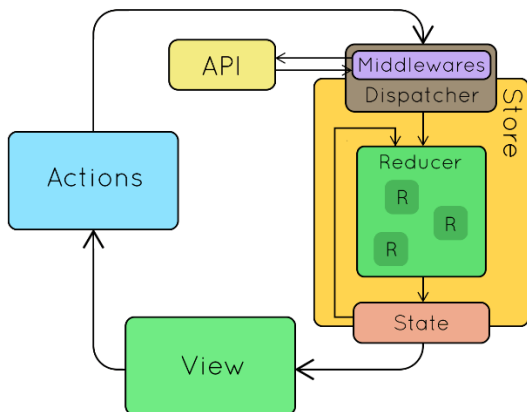
♦ Client : présentation + applicatif



Architecture du client :

Redux reprend l'architecture de Flux mais il omet la complexité inutile.

- Redux n'a pas de concept DISPATCHER.
- Redux n'a que STORE au lieu de plusieurs STORE comme le Flux.
- Les objets Action seront recus et gérés directement par STORE.



VIEW PROVIDER: Représente un View Framework pour inscrire avec STORE. Dans lequel, View Framework peut être React ou Angular,...

ACTION: un objet pur créé pour stocker les informations relatives à l'événement d'un utilisateur (cliquez sur l'interface, ...). Il inclut les informations telles que: le type d'action, l'heure de l'événement, l'emplacement de l'événement, ses coordonnées et quel state qu'il vise à modifier.

STORE: Gère l'état de l'application et a la fonction dispatch(action).

Conception BDD

MCD : Modèle Conceptuel de Données.

MOD : Modèle Organisationnel de Données.

MLD : Modèle Logique de Données.

MRD : Modèle Relationnel de Données.

MPD : Modèle Physique de Données.

I. Étapes du MCD détaillé :

Étape 1 : Identification des entités.

- USER
- CUSTOMER
- DOCUMENT
- AGENCE

Étape 2 : Lister les propriétés des entités.

- AGENCE : name, code d'agence.
- USER : name, firstname, password, email, **picture**.
- CUSTOMER : name, firstname, email, dateCreation, comment.
- DOCUMENT : name, type, **pdf**.

attributs dont on ne sait pas encore comment s'y prendre

Étape 3 : Identification des données uniques.

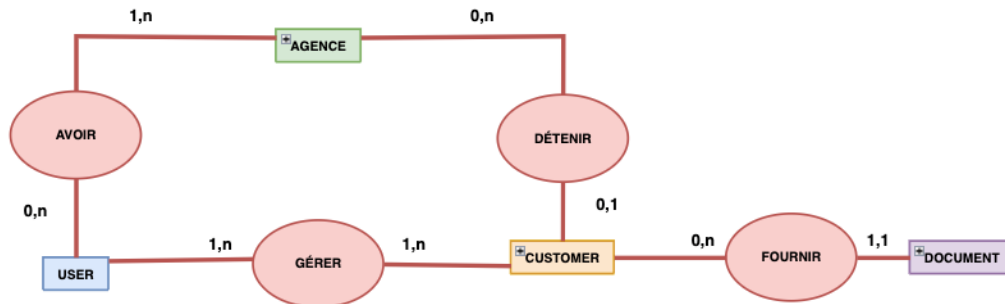
- AGENCE : **ID**, name, codeAgence.
- USER : **ID**, name, firstname, password, email, picture.
- CUSTOMER : **ID**, name, firstname, email, dateCreation, comment.
- DOCUMENT : **ID**, name, type, pdf.

Étape 4 : Rechercher les associations entre les entités.

- 1 - Une agence **A** un ou plusieurs users.
- 2 - Un user peut **AVOIR** zéro ou plusieurs agences.
- 3 - Un customer **EST GÉRER** par un ou plusieurs users.
- 4 - Un user peut **GÉRER** un ou plusieurs customers.
- 5 - Une agence **DÉTIENT** zéro ou plusieurs customers.

- 6 - Un customer **EST DÉTENU** par zéro ou plusieurs agences.
- 7 - Un customer **FOURNIS** un ou plusieurs documents.
- 8 - Un document peut **ÊTRE FOURNIS** par un seul customer.

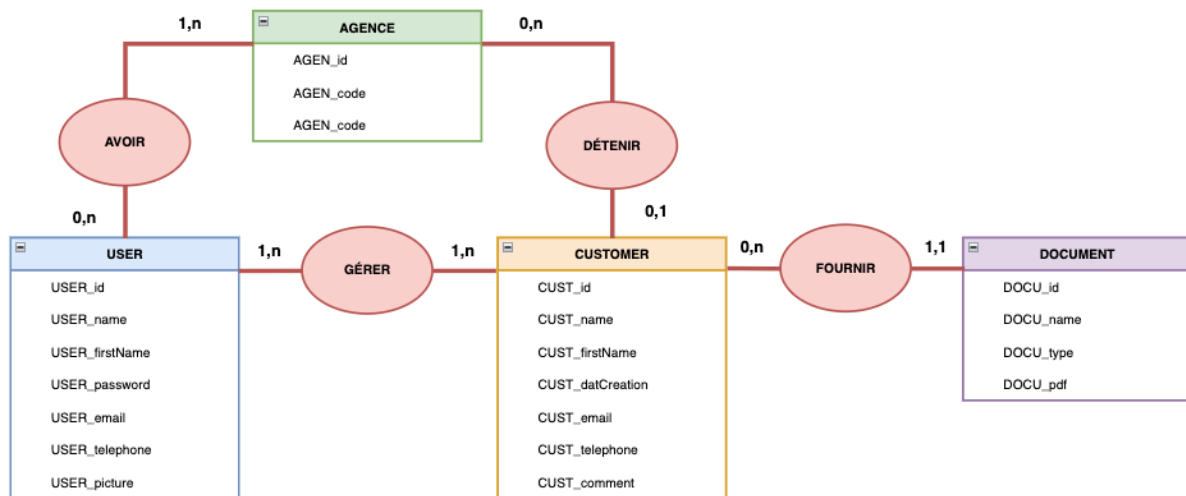
Étape 5 : Identification des cardinalités.



Étape 6 : Normalisation du schéma.

- Normalisation des entités : RAS (il n'y a pas de cardinalité 0,n vers 0,n donc pas de création d'une nouvelle table).
- Normalisation des noms : Entités au singulier, nom des attributs corrects et transparents.
- Normalisation des attributs : Ajout du nom de l'entité (trois premières lettres) au début de l'attribut. Il peut y avoir une ambiguïté entre les tables (ex : plusieurs tables comportent l'attribut name).
- Normalisation des associations : RAS.
- Normalisation des cardinalités : RAS.

Étape 7 : MCD final.

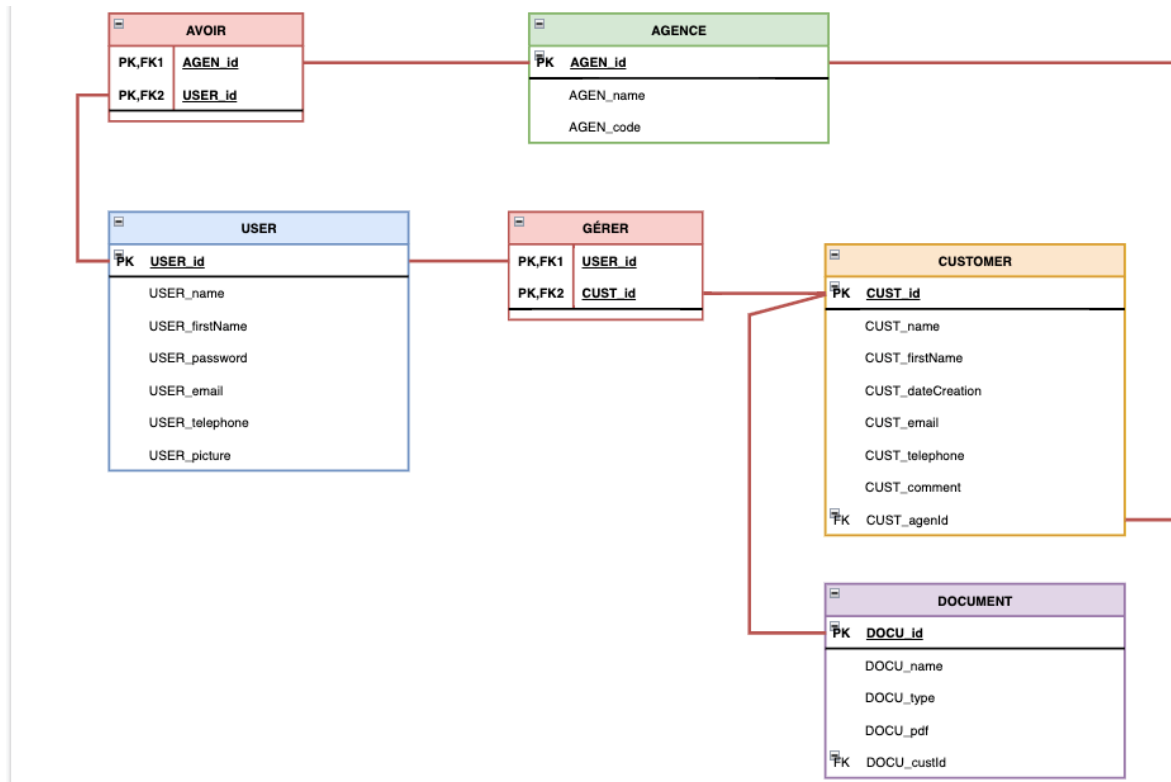


. PASSAGE DU MCD AU MRD -

II. Transformation d'une entité

- USER (USER_id, USER_name, USER_firstName, USER_password, USER_email, USER_telephone, USER_picture)
 - AGENCE (AGEN_id, AGEN_name, AGEN_code)
 - AVOIR (AGEN_id, USER_id)
 - GÉRER (USER_id, CUST_id)
 - CUSTOMER (CUST_id, CUST_name, CUST_firstName, CUST_dateCreation, CUST_email, CUST_telephone, CUST_comment, CUST_userId = USER_id)
 - DOCUMENT (DOCU_id, DOCU_name, DOCU_type, DOCU_pdf, DOCU_custId = CUST_id)
- * Clé étrangère + référence

. PASSAGE DU MRD AU MLD -



. PASSAGE DU MLD AU MPD -

```

USER (
  USER_id : ENTIER,
  USER_name : CHAÎNE NOT
  NULL,
  USER_firstName : CHAÎNE
  NOT NULL,
  USER_password : CHAÎNE
  NOT NULL,
  USER_email : CHAÎNE NOT
  NULL UNIQUE,
  USER_telephone CHAÎNE,
  USER_picture CHAÎNE
)
    
```

```

AGENCE(
  AGEN_id : ENTIER UNIQUE,
  AGEN_name : CHAÎNE,
  AGEN_code : ENTIER
)
    
```

```

CUSTOMER(
  CUST_id : ENTIER,
  CUST_name : CHAÎNE,
  CUST_firstName : CHAÎNE,
  CUST_dateCreation : DATE
  NOT NULL,
  CUST_email : CHAÎNE
  UNIQUE,
  CUST_telephone : ENTIER,
  CUST_comment : TEXT,
  CUST_userId : ENTIER NOT
  NULL UNIQUE
)
    
```

```

AVOIR(
  AVOIR_agenId : ENTIER,
  AVOIR_userId : ENTIER
)
    
```

```

DOCUMENT(
  DOCU_id : ENTIER,
  DOCU_name : CHAÎNE,
  DOCU_type : CHAÎNE,
  DOCU_pdf : CHAÎNE,
  DOCU_custId : ENTIER NOT
  NULL UNIQUE
)
    
```

```

GÉRER(
  GERER_custId : ENTIER,
  GERER_doculd : ENTIER
)
    
```

* TYPE DE DONNÉES * NOT NULL * UNIQUE

Conception Back-end

Api Plateforme Symfony



Pour la construction du back-end nous avons fait les choix techniques de coupler Symfony à API Plateforme.

Symfony :

Dans ce projet, nous utiliserons symfony pour gérer plusieurs aspects du projet, la connexion avec token, et la sécurité.

Tout d'abord l'utilisation de symfony était comme dit plus haut une exigence contrainte dans le développement de l'application, mais symfony a beaucoup d'avantages dont le fait que les projets créés avec cette plateforme sont hautement extensibles en raison de l'architecture modulaire. L'utilisation de bundles et de composants en fait une solution brillante pour les sites Web et les applications de toute taille et complexité.

Et Bien qu'il ne soit pas facile de l'apprendre, on peut toujours trouver une bonne documentation, des cours officiels et le soutien de la communauté qui est très présente.

Api Plateforme :

Dans ce projet nous utiliserons Api plateforme pour créer nos points d'entrée et notre API Rest

Nous avons utilisé API-Platform car c'est une distribution Symfony qui permet de créer rapidement et simplement de puissantes API REST, au moyen de quelques annotations sur nos Entités. Elle fait partie des distributions officielles de Symfony et on trouve donc une doc à jour avec les dernières versions de Php.

L'aspect de la documentation au format OpenAPI (ex Swagger) est aussi entré en jeu pour notre décision d'utiliser API Plateforme, en effet cela semblait une bonne manière de gagner du temps pour le développement du projet.

Création des routes :

L'utilisation d'API nous a permis de créer directement une API Rest et de gérer les points plus facilement. Api plateforme crée directement les « controller »

API-Platform a généré automatiquement pour chaque entité 6 routes par défaut:

- GET /customers : liste toutes les customers
- POST / customer : création de customer
- GET / customer /{id} : affiche les détails d'un customer en particulier
- DELETE / customer /{id} : supprime un customer en particulier
- PATCH / customer /{id} : édite un customer en particulier
- PUT / customer/{id} : remplace un customer

Comment ça marche ? :

API-Platform vous permet d'activer et/ou de désactiver une opération en particulier, ainsi que de customiser chacune des actions. Les informations qui sont remontées lors d'un GET peuvent être aisément configurées par le biais des groupes de sérialisation du Serializer Symfony. De même, les annotations du validateur sur nos propriétés permettent de remonter les erreurs lors des opérations POST, comme on le ferait pour un formulaire classique.

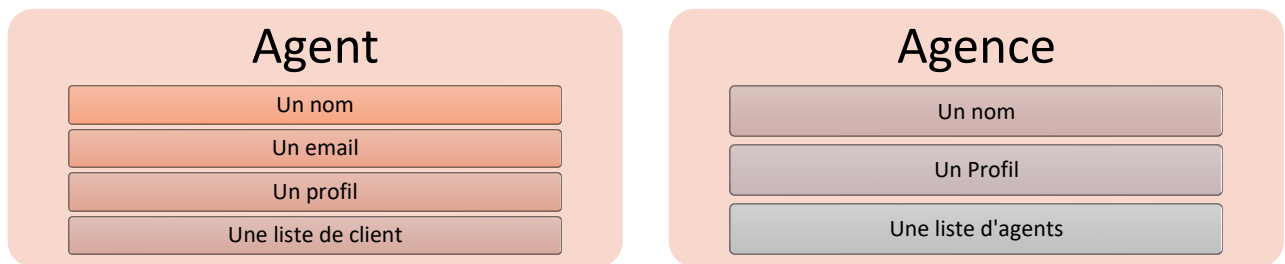
```
/**
 * @ORM\Entity(repositoryClass=CustomerRepository::class)
 * @ApiResource(
 *     attributes={
 *         "maximum_items_per_page"=10,
 *         "security"="is_granted('ROLE_USER')"
 *     },
 *     normalizationContext={"groups"={"read:collection", "read:Customer"}},
 *     denormalizationContext={"groups"={"write:Customer:item"}},
 *     itemOperations={
 *         "get"={
 *             "normalization_context"={"groups"={"read:item", "read:Customer"}}
 *         },
 *         "put",
 *         "delete",
 *         "patch"
 *     }
 * )
 * @ApiFilter(SearchFilter::class, properties={
 *     "name": "partial",
 *     "firstname": "partial"
 * })
 */
```

En déclarant ici que mon système de **<Customer>** et une ressource d'API, ça va créer automatiquement les points d'entrée rest, (@Apiressources())

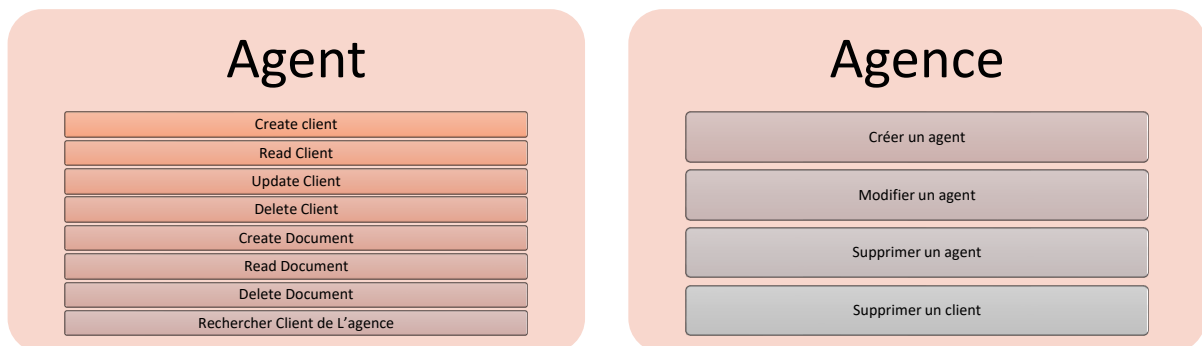
Par défaut ça va utiliser le format REST mais cela fonctionne aussi avec GraphQL

Spécifications fonctionnelles

Caractéristiques utilisateurs



Actions utilisateurs



Les données

Gestion des données métier :

Représentent les données liées au cœur de métier de SpiesImmo, elles ont vocation à être persistées.

Document :

- Un nom
- Une date de création
- Une date de parution
- un contenu
- Une catégorie

Client :

- Un nom
- Un prénom
- Un email
- Une description
- Des documents
- Un agent

Données calculées :

Les indicateurs de performance sont calculés à partir des données métiers et des Utilisateurs. Ils représentent une situation à un instant donné. Ils ne sont pas persistés.

Indicateur performance agent :

- Un nombre de clients par agent

Indicateur performance entreprise :

- Un nombre de clients total par agence

Sécuriser l'accès à l'application aux personnes accréditées :

L'authentification est réalisée via un couple login/mot de passe qui aura été défini à la création du compte utilisateur.

Respecter la réglementation RGPD pour les informations des agents :

Nous ne collectons, sur les agents, seulement les données vraiment nécessaires :

- Nom
- Prénom
- email

L'agent doit signer un formulaire afin qu'il autorise le traitement par la société des données personnelles nous permettant de l'enregistrer dans notre annuaire.

Lors de la suppression de son compte, les informations de l'agent ou de l'agence persisteront pendant 6 mois.

Quand l'utilisateur cesse le paiement sur l'application pendant plus d'un an, ses informations seront automatiquement oubliées.

Réalisation Back-end

La sérialisation

Mise en place :

Tout comme les autres composants Symfony et API Platform, le composant Sériailzer peut être configuré à l'aide d'annotations, XML ou YAML.

```
# api/config/packages/framework.yaml
framework:
    serializer:
        mapping:
            paths: ['%kernel.project_dir%/config/serialization']
```

Utilisation des groupes de sérialisation :

Il est simple de spécifier les groupes à utiliser dans le système d'API :

- Les attributs qui se trouveront dans le groupe du normalizationContext seront accessible en mode lecture (GET)
- Les attributs qui se trouveront dans le groupe du denormalizationContext seront accessible en mode écriture (POST, PUT, PATCH)
- Les attributs qui auront les deux groupes seront accessible en mode lecture et écriture
- Les attributs qui n'auront aucun des groupes ne seront pas pris en compte

Ajoutez les attributs de contexte de normalisation et de contexte de dénormalisation à la ressource et spécifier les groupes à utiliser. Ici, nous ajoutons read et write, respectivement.

```
/**
 * @ORM\Entity(repositoryClass=CustomerRepository::class)
 * @ApiResource(
 *     attributes=
 *         normalizationContext={"groups"={"read:Customer"}},
 *         denormalizationContext={"groups"={"write:Customer:item"}},
 */
```

```
/**
 * @ORM\Column(type="string", length=255, nullable=true)
 * @Groups({"read:item", "write:Customer:item"})
 */
private $name;
```


Les opérations API Plateforme :

Création d'un model

Notre premier modèle en `api/src/Entity/Customer.php`.

Ce modèle sera responsable du stockage des éléments de notre liste client.

Après avoir confirmé que les modèles ont été créés, il est maintenant temps de créer des points de terminaison avec des opérations CRUD. Pour ce faire, nous devons marquer la classe que nous avons créée à l'aide de l' `@ApiResponse` annotation. Notre classe ressemblera à :

```
/**
 * @ORM\Entity(repositoryClass=CustomerRepository::class)
 * @ApiResponse()
 */
class Customer
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     * @Groups({"read:item", "write:Customer:item"})
     */
    private $name;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     * @Groups({"read:collection", "read:item", "write:Customer:item"})
     */
    private $firstname;
```

Public / Private :

Définir des attributs privés afin que la personne qui utilise la classe ne puisse pas modifier les attributs de la classe comme il le souhaite.

Cependant, lorsque ces attributs sont privés, ils sont « inaccessibles » depuis une autre classe. Mais il existe un moyen de les afficher via une méthode `get...()` et il est tout de même possible de modifier les attributs via une méthode `set`.

```
//Creation constructeur
public function __construct()
{
    $this->datecreation = new \DateTime();
    $this->updateAt = new \DateTime();
}

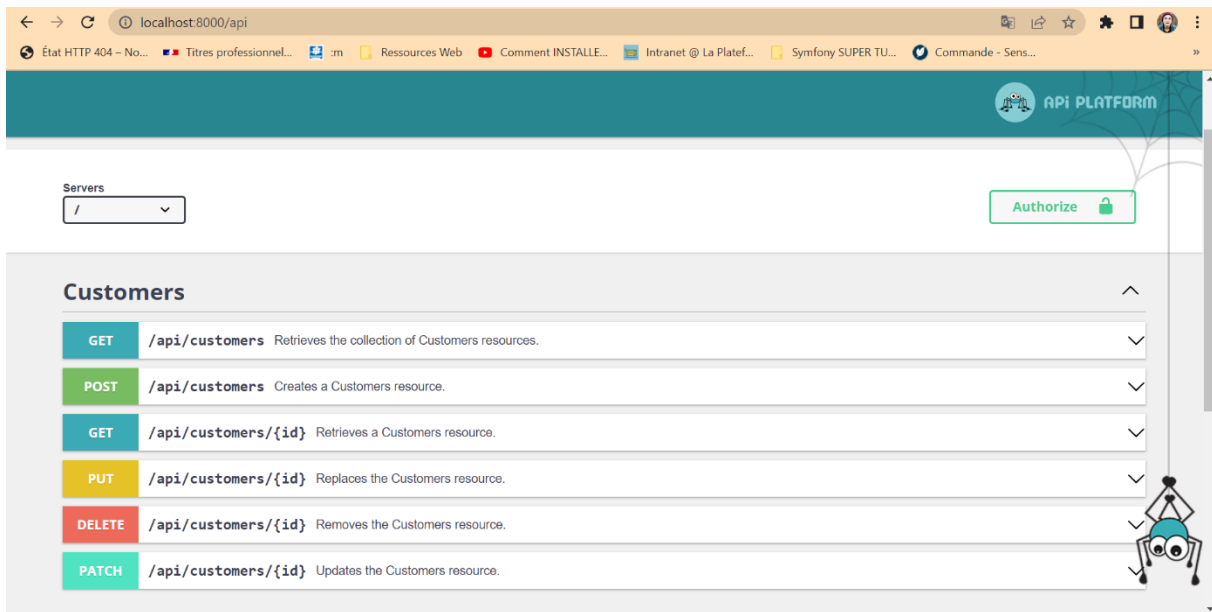
public function getId(): ?int
{
    return $this->id;
}

public function getName(): ?string
{
    return $this->name;
}

public function setName(?string $name): self
{
    $this->name = $name;
```

Le concept

Le concept fondamental de toute API REST c'est la Ressource. Une ressource est un objet avec un type, des données associées, des relations avec d'autres ressources et un ensemble de méthodes qui opèrent dessus. La ressource est similaire à un objet en programmation orienté objet, avec la différence importante que seules quelques méthodes standard sont définies pour la ressource (correspondant aux méthodes HTTP GET, POST, PUT et DELETE standard), tandis qu'un objet a généralement de nombreuses méthodes.



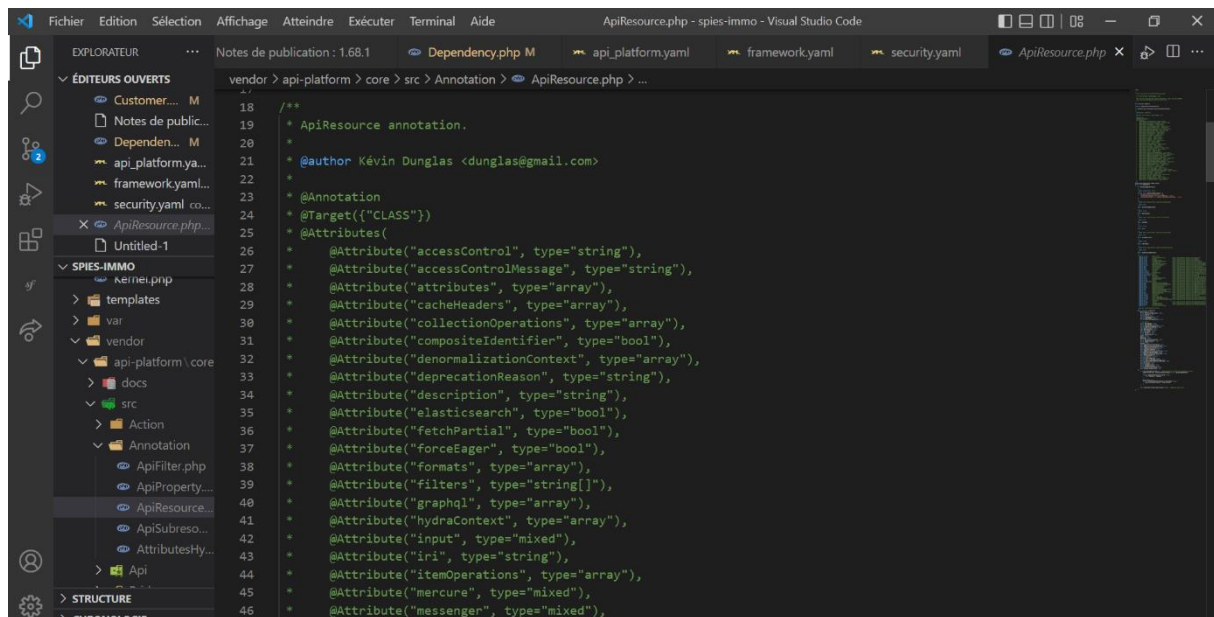
Modification de l'entité à l'aide d'attribut dans Api Plateforme

```
/**
 * @ORM\Entity(repositoryClass=CustomerRepository::class)
 * @ApiResponse(
 *     attributes={
 *         "maximum_items_per_page"=10,
 *         "security"="is_granted('ROLE_USER')",
 *     },
 *     normalizationContext={"groups"={"read:Customer"}},
 *     denormalizationContext={"groups"={"write:Customer:item"}},
 *     itemOperations={
 *         "get"={
 *             "normalization_context"={"groups"={"read:item", "read:Customer"}}
 *         },
 *         "put",
 *         "delete",
 *         "patch"
 *     }
 * )
```

ApiRessource et Annotation :

A l'initialisation d'api plateforme, une documentation sur les annotations et fonctionnalités se génèrent automatiquement.

Notre documentation est basée sur une version antérieure à php8, nous sommes encore avec des annotations plutôt que des attributs, que la version 8 de php utilise.



Connexion avec JwtToken

Type d'authentification :

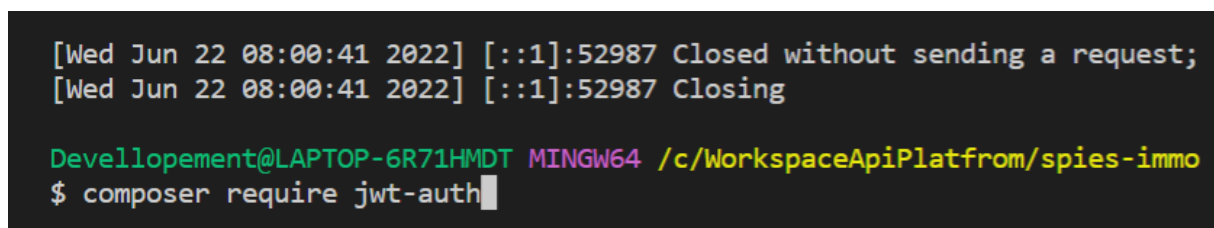
Authentification JSON

Authentification Form HTML

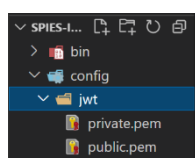
Authentification JWT

API Platform permet d'ajouter facilement une authentification basée sur JWT à votre API à l'aide de LexikJWTAuthenticationBundle .

On commence par installer le bundle :



Ensuite on génère les clefs privée et publique



Puis créer une UserClasse

```
class User implements UserInterface, PasswordAuthenticatedUserInterface

/**
 * @ORM\Id
 * @ORM\GeneratedValue
 * @ORM\Column(type="integer")
 * @Groups("user:read")
 */
private $id;

/**
 * @ORM\Column(type="string", length=180, unique=true)
 * @Groups({"user:read", "user:write"})
 */
private $email;

/**
 * @ORM\Column(type="json")
 * @Groups("user:read")
 */
private $roles = [];

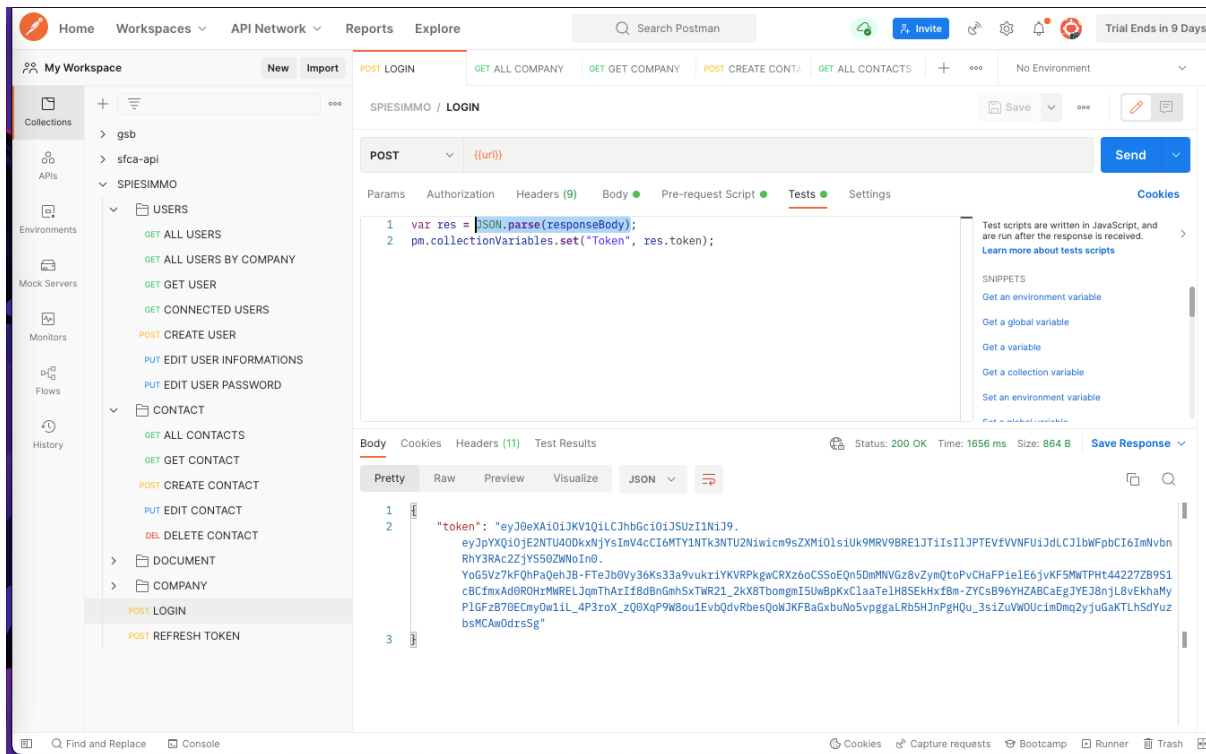
/**
 * @var string The hashed password
 * @ORM\Column(type="string")
 */
private $password;
```

Et, on termine par mettre à jour la configuration de sécurité :

```
providers:
    # used to reload user from session & other features (e.g. switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: email
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    login:
        pattern: ^/api/login
        stateless: true
        json_login:
            check_path: /api/login
            username_path: email
            password_path: password
            success_handler: lexik_jwt_authentication.handler.authentication_success
            failure_handler: lexik_jwt_authentication.handler.authentication_failure
    api:
        pattern: ^/api/
        stateless: true
        provider: app_user_provider
        guard:
            authenticators:
                - lexik_jwt_authentication.jwt_token_authenticator
```

PostMan :

On écrit une collection postman avec des variables de connexion, ensuite on crée des préscripts afin d'automatiser l'URL à chaque envoi de requête. On récupère également automatiquement le token.



Grace à Postman, on peut publier un swagger de l'api, qui va permettre d'avoir une documentation simple pour le front-end.

L'onglet Documentation du générateur d'API fournit un emplacement unique pour afficher, créer et gérer toute la documentation de votre API. Postman génère automatiquement des documents API pour tout schéma OpenAPI 3.0 défini dans API Builder. Vous pouvez également ajouter une documentation détaillée à n'importe quelle API en générant une collection à partir de l'API ou en créant un lien vers une collection existante.

The image displays two screenshots of the Postman application interface, showing the 'Variables' and 'Pre-request Script' tabs for a collection named 'SPIESIMMO'.

Top Screenshot: Variables Tab

The 'Variables' tab is active, showing a table of variables specific to the collection and its requests. The table has columns: VARIABLE, INITIAL VALUE, CURRENT VA, Persist All, and Reset All.

VARIABLE	INITIAL VALUE	CURRENT VA	Persist All	Reset All
<input checked="" type="checkbox"/> url	https://sfca-api...	https://sfca-api.tech/api/contact		
<input checked="" type="checkbox"/> Token		eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiU9...		

Below the table, there is a text box with the message: "Use variables to reuse values and protect sensitive data. Store sensitive data in variable type secret to keep its values masked on the screen. Learn more about variable type. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values."

Bottom Screenshot: Pre-request Script Tab

The 'Pre-request Script' tab is active, showing a JavaScript snippet for setting a collection variable:

```
1 pm.collectionVariables.set("url", "https://sfca-api.tech/api/companies")
```

The right sidebar shows the 'Send' button and a 'Cookies' section. The bottom status bar indicates the request was successful with a status of 200 OK, time of 117 ms, and size of 4.16 KB.

Conception front-end

Choix des technos

React native est l'un des frameworks JavaScript les plus populaires dans le monde aujourd'hui. Il est célèbre pour son utilité dans la création d'applications natives sur les plateformes Android et iOS. React Native trouve ses racines dans React ; c'est une bibliothèque JavaScript que Facebook a créée pour concevoir l'interface des applications mobiles. React Native peut être combiné avec JavaScript pour créer des applications ayant des fonctionnalités similaires à celles des applications natives. Parce que les codes React Native sont partageables, c'est une excellente plateforme pour le développement d'applications multiplateformes. Les applications React Native exploite une combinaison de technologies telles que JavaScript, JSX, le balisage de type XML, et bien d'autres encore. Le pont React Native gère le rendu des API en Swift et Java pour la création d'applications iOS et Android, respectivement. Le rendu des applications de ce framework utilise l'interface utilisateur mobile plutôt que WebView. Il donne au développeur une interface pour interagir avec les API de la plate-forme de l'application. C'est pourquoi React Native peut intégrer des fonctionnalités natives telles que l'accès au GPS et à l'appareil photo de l'appareil. Actuellement, il crée des applications natives pour les plates-formes Android et iOS et peut prendre en charge des applications supplémentaires.

AVANTAGES :

1. Développement rentable.
2. Livraison plus rapide des projets d'application.
3. Utiliser JavaScript.
4. Nécessite des équipes plus petites.
5. Avantage de l'Open-Source.
6. Fonction de rechargement à chaud.
7. Une communauté de développeurs active.
8. Excellente performance de l'application.
9. Apparence de type native.
10. La conception modulaire.

INCONVÉNIENTS :

1. Les défis du débogage et de la compatibilité.
2. Vous avez toujours besoin de développeurs natifs.
3. Dépend de Facebook.
4. La gestion de la mémoire n'est pas exceptionnelle.
5. Adoption lente des dernières fonctions.
7. Défis en matière de sécurité avec JavaScript.
8. Problèmes de performance par rapport aux caractéristiques.
9. Composants de développement tiers.

redux :

Redux est une librairie externe à React qui se focalise sur la gestion de données dans des applications JavaScript. Il est important de noter que Redux peut également être utilisé avec d'autres applications web réalisées sans React, mais elle est régulièrement associée à ce framework.

Arborescence :

Pourquoi expo ?

Expo embarque de nombreux outils utiles et des librairies natives pour React Native. Il gère aussi la mise à jour de ces librairies. C'est donc un moyen de démarrer facilement et rapidement son projet.

Expo a l'avantage d'être très simple à configurer et il simplifie grandement la vie des développeurs, à tous les stades du projet :

- Développement : vous vous contentez d'écrire le code, Expo s'occupe du build des applications.
- Test : Expo vous permet de tester et de faire tester vos applications sans avoir à les publier sur les stores. Il suffit d'installer l'application Expo Go et de scanner un QR code pour pouvoir tester une application en développement.
- Publication et mise à jour sur les stores : Expo simplifie la publication des applications sur les stores Apple et Google, et permet de les mettre à jour sans passer par la validation des stores (Over the Air updates).

— Expo : les inconvénients

Avant de décider d'utiliser Expo, il faut bien être conscient de ses limites.

Tout d'abord, Expo prend la main sur les processus de build et publish qui ne sont en théorie pas liés à votre framework. Avec Expo ça sera forcément lié à React Native et vous ne pourrez pas capitaliser votre expérience sur d'autres frameworks.

Ensuite, Expo est livré comme on l'a vu avec de nombreuses librairies React Native. C'est intéressant, certes... sauf que même si vous ne les utilisez pas toutes, votre application les chargera quand même.

C'est pourquoi le poids des applications générées par Expo est très élevé : 20 Mo minimum pour iOS, 15 Mo minimum pour Android.

Un autre inconvénient majeur, c'est que malgré toutes ces librairies, il manque des fonctionnalités cruciales à Expo : par exemple, votre application ne pourra pas intégrer d'achats in app, ni utiliser le Bluetooth, et si vous voulez afficher une carte vous serez contraints d'utiliser Google Maps. Ces limites peuvent être carrément bloquantes.

Terminons avec un inconvénient qu'il faut connaître : une application buildée avec Expo ne peut pas être fournie à des enfants de moins de 13 ans car elle embarque des librairies Facebook (même si on ne les utilise pas). Bon à savoir si vous avez un projet qui s'adresse au grand public.

ReactNative IOS et Android

Sur iOS, est soutenu par du code natif qui stocke de petites valeurs dans un dictionnaire sérialisé et des valeurs plus grandes dans des fichiers séparés. Sur Android, utilisera RocksDB ou SQLite en fonction de ce qui est disponible.

Le code JavaScript est une façade qui fournit une API JavaScript claire, des objets réels et des fonctions non multiples.

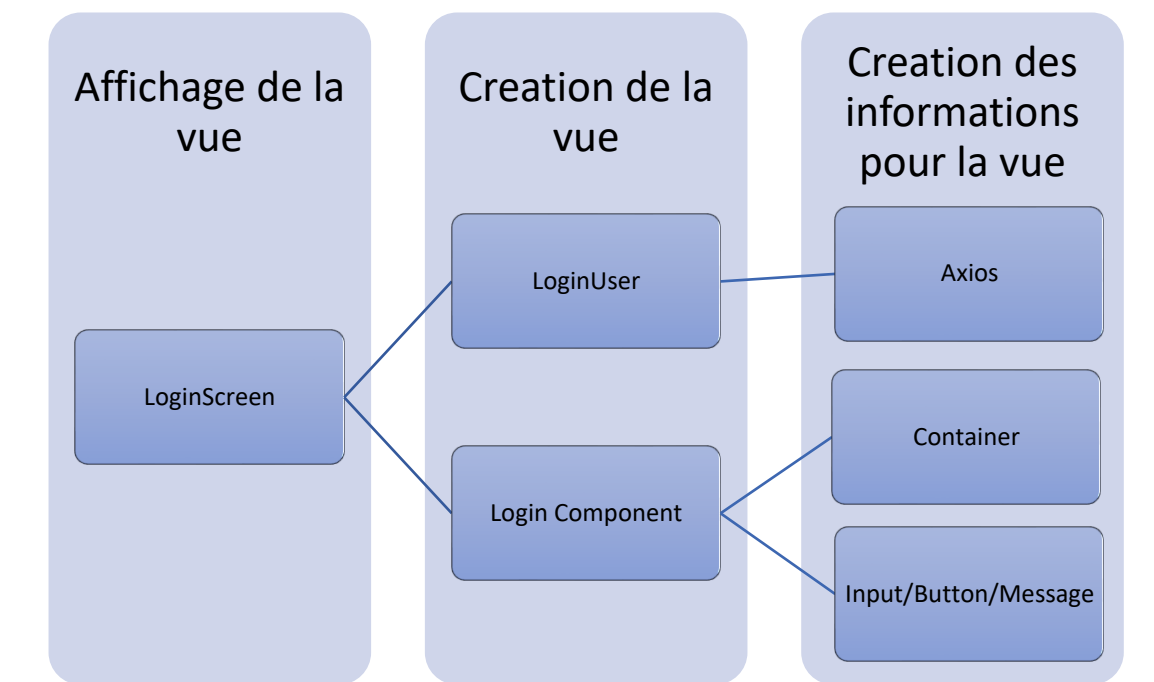
Nous avons rencontré beaucoup de difficultés au niveau de l'asyncstorage, surtout pour les fonctionnalités touchant directement à l'environnement du téléphone.

En effet async storage comme dit plus haut utilise différentes méthodes selon le système. Pour gérer ce problème, nous avons modifié le code dans les bundler, en effet avec expo nous n'avons pas rencontré de problème réels car nous utilisons les commandes comme "expo start" qui ne correspond pas à un environnement de production.

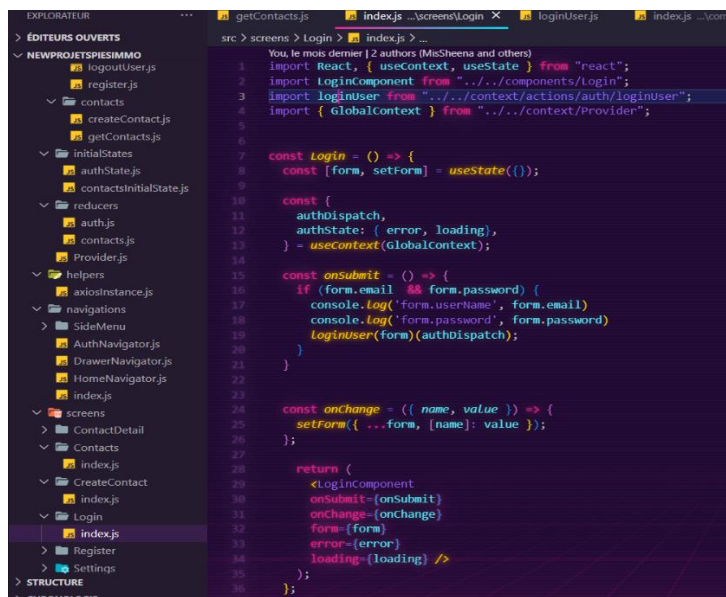
Réalisation Front-end:

Formulaire de connexion

Comment c'est construit ?



⇒ Login Screen et l'affichage de la vue du formulaire de connexion :



```
1 import React, { useContext, useState } from "react";
2 import LoginComponent from "../../components/Login";
3 import loginUser from "../../context/actions/auth/loginUser";
4 import { GlobalContext } from "../../context/Provider";
5
6
7 const Login = () => {
8   const [form, setForm] = useState({});
9
10  const {
11    authDispatch,
12    authState: { error, loading },
13  } = useContext(GlobalContext);
14
15  const onSubmit = () => {
16    if (form.email && form.password) {
17      console.log('form:username', form.email);
18      console.log('form:password', form.password);
19      loginUser(form)(authDispatch);
20    }
21  }
22
23  const onChange = ({ name, value }) => {
24    setForm({ ...form, [name]: value });
25  };
26
27  return (
28    <LoginComponent
29      onSubmit={onSubmit}
30      onChange={onChange}
31      form={form}
32      error={error}
33      loading={loading} />
34  );
35 }
```

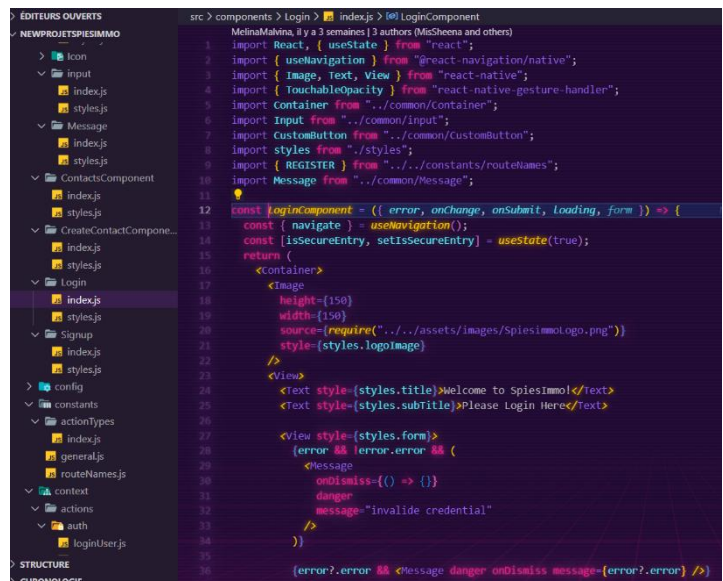
Il dépend de LoginComponent, comme son nom l'indique c'est ici où sont créés les champs visuels qui serviront à construire la vue. On y importe plusieurs classes créées en amont, comme Input, ou Custom Button.

Effectivement ici, nous pouvons voir dans les imports que nous utilisons par exemple « Text » importé directement de la bibliothèque ReactNative, mais nous n'y importons pas directement input, ou button.

⇒ LoginComponent Création de la vue :

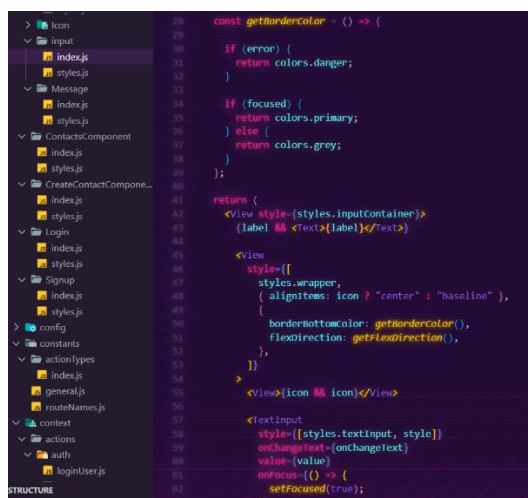
Comme mentionnée plus haut LoginComponent est constitué de plusieurs classes par exemple on vient définir la taille du container **de dans** container.

Nous avons aussi créé les composants input, et button pour pouvoir les rappeler au besoin. Cela nous permet d'avoir la main sur la réaction des composants, de comprendre leur construction et de leur donner des méthodes qui pourront être rappelées au besoin.

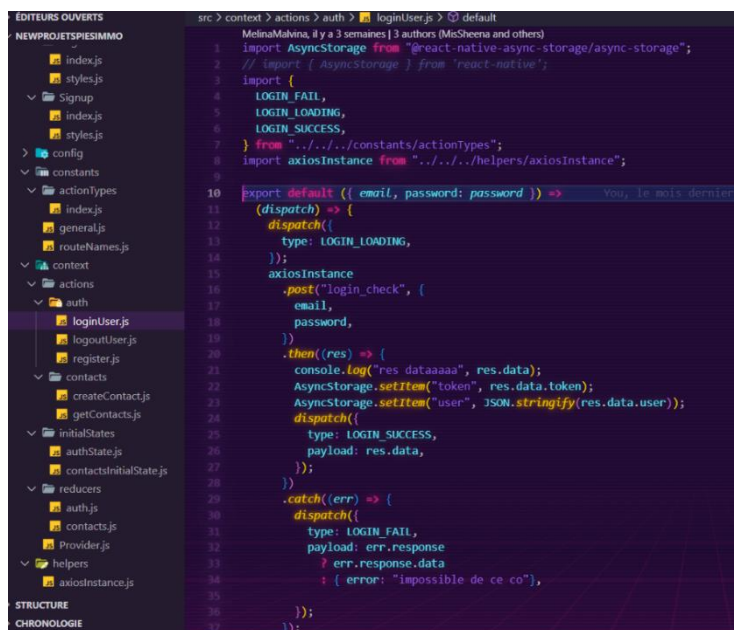


```
1 import React, { useState } from "react";
2 import { useNavigation } from "@react-navigation/native";
3 import { Image, Text, View } from "react-native";
4 import { TouchableOpacity } from "react-native-gesture-handler";
5 import Container from "../../common/Container";
6 import Input from "../../common/Input";
7 import CustomButton from "../../common/CustomButton";
8 import styles from "../../styles";
9 import { REGISTER } from "../../constants/routeNames";
10 import Message from "../../common/Message";
11
12 const LoginComponent = ({ error, onChange, onSubmit, loading, form }) => {
13   const { navigate } = useNavigation();
14   const [isSecureEntry, setIsSecureEntry] = useState(true);
15   return (
16     <Container>
17       <Image
18         height={150}
19         width={150}
20         source={require("../assets/images/spiesimmo.png")}
21         style={styles.logoImage}
22       />
23       <View>
24         <Text style={styles.title}>Welcome to Spiesimmo</Text>
25         <Text style={styles.subtitle}>Please Login Here</Text>
26       </View>
27       <View style={styles.form}>
28         <Input
29           error={error}
30           errorMessage={error}
31           onDismiss={() => {}}
32           danger
33           message="invalid credential"
34         />
35       </View>
36       <CustomButton
37         error={error}
38         danger
39         onDismiss={message={error?.error}} />
40     </Container>
41   );
42 }
```

⇒ Input nous donnera le rendu d'un input visuel :



⇒ LoginUser pour la requête :

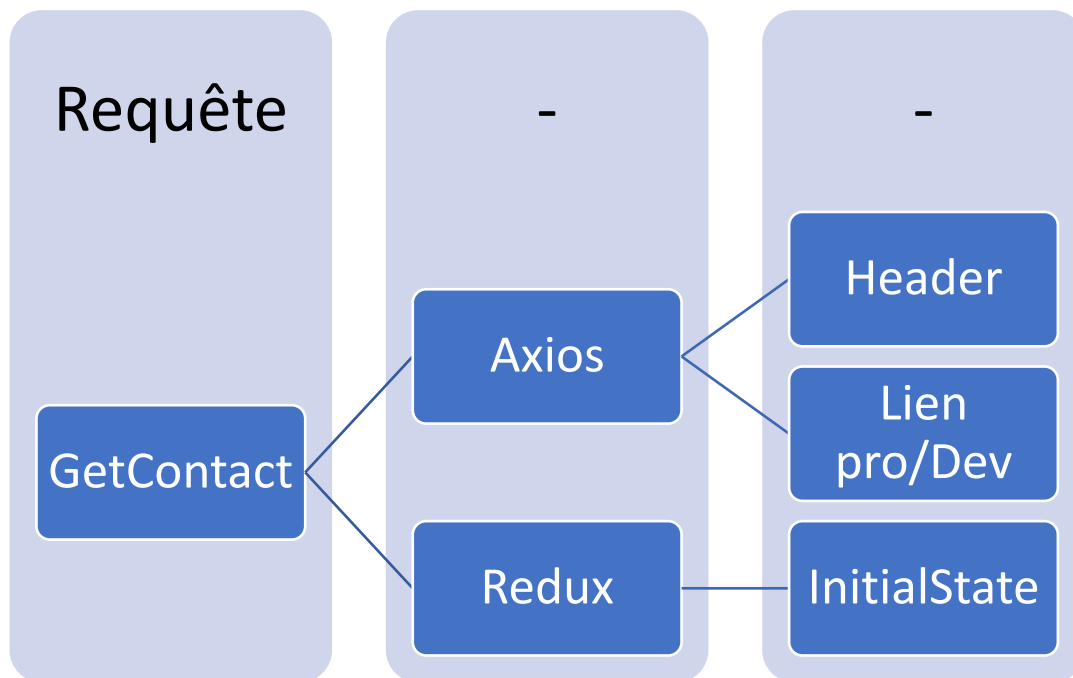


Login Screen qui dépendra aussi LoginUser dans lequel on retrouve la requête axios
On y utilise asyncstorage pour garder le token.

Comme les différents cas ont été paramétrés en amont dans le reducer, on peut les gérer en rappelant uniquement success, loading.

Get des contact avec axios

Comment c'est construit ?

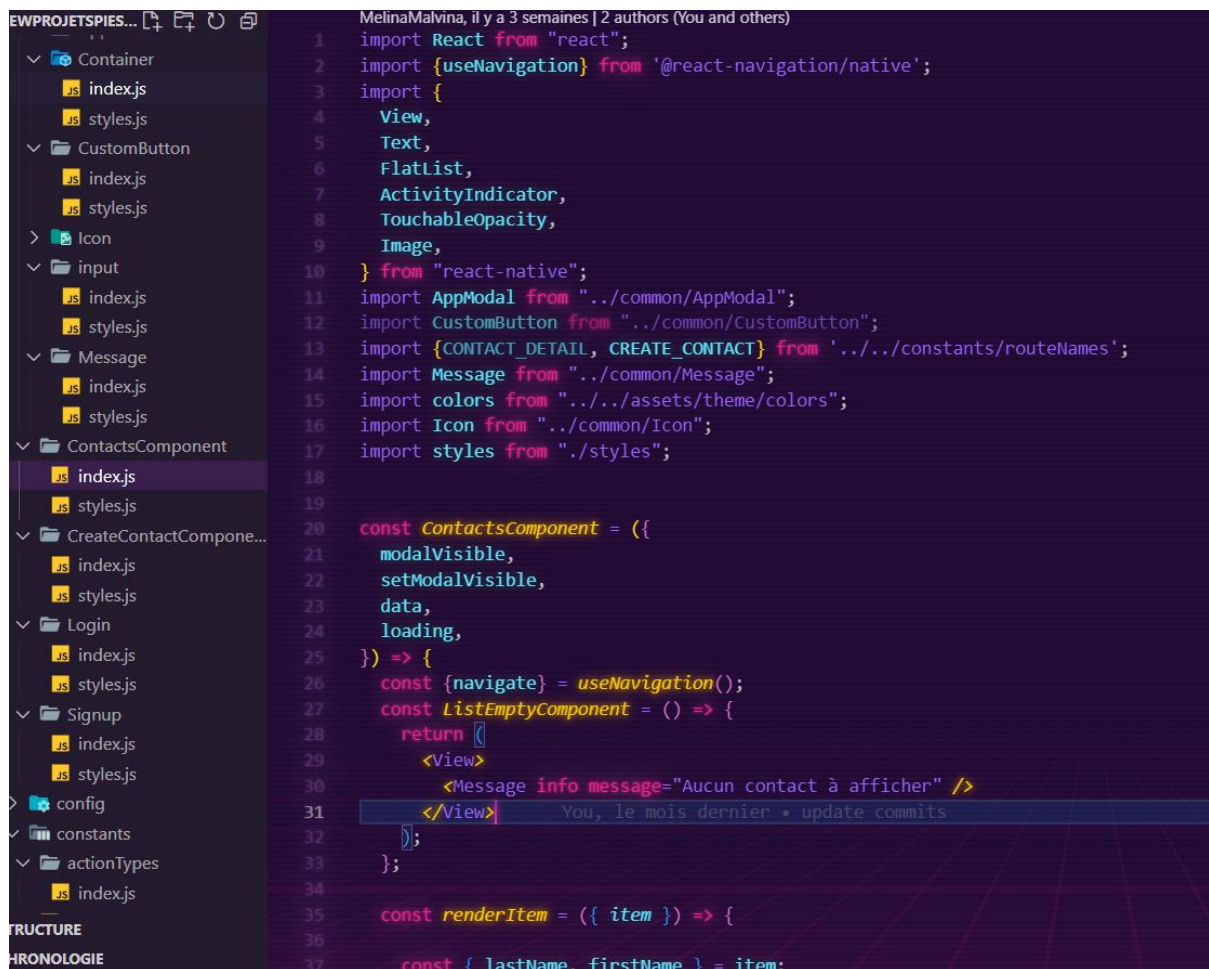


ScreenContacts

- Comme pour le formulaire de connexion nous importons nos propres composants, ainsi que d'autres appartenant à la bibliothèque React Native

ContactsComponent

- On lui importe les "libraries" nécessaires à l'affichage de la vue de la liste
- Il utilisera le style global de l'application, mais aussi les données reçues pour construire chaque card de contact.



```
1 import React from "react";
2 import {useNavigation} from '@react-navigation/native';
3 import {
4   View,
5   Text,
6   FlatList,
7   ActivityIndicator,
8   TouchableOpacity,
9   Image,
10 } from "react-native";
11 import AppModal from "../common/AppModal";
12 import CustomButton from "../common/CustomButton";
13 import {CONTACT_DETAIL, CREATE_CONTACT} from '../../constants/routeNames';
14 import Message from "../common/Message";
15 import colors from "../../assets/theme/colors";
16 import Icon from "../common/Icon";
17 import styles from "../styles";
18
19
20 const ContactsComponent = ({
21   modalVisible,
22   setModalVisible,
23   data,
24   loading,
25 }) => {
26   const {navigate} = useNavigation();
27   const ListEmptyComponent = () => {
28     return (
29       <View>
30         <Message info message="Aucun contact à afficher" />
31       </View>
32     );
33   };
34
35   const renderItem = ({ item }) => {
36     const { lastName, firstName } = item;
```

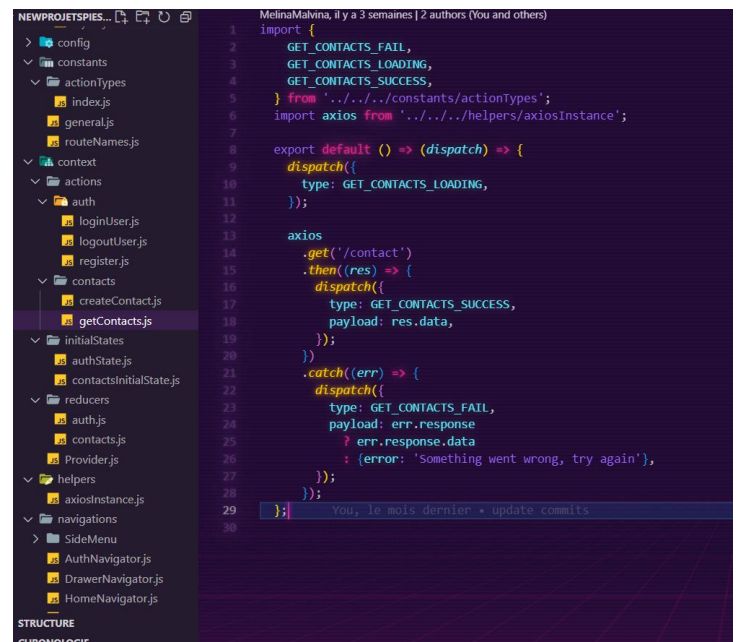

GetContacts

On gère les cas , de success, loading, et fail grâce à notre reducer

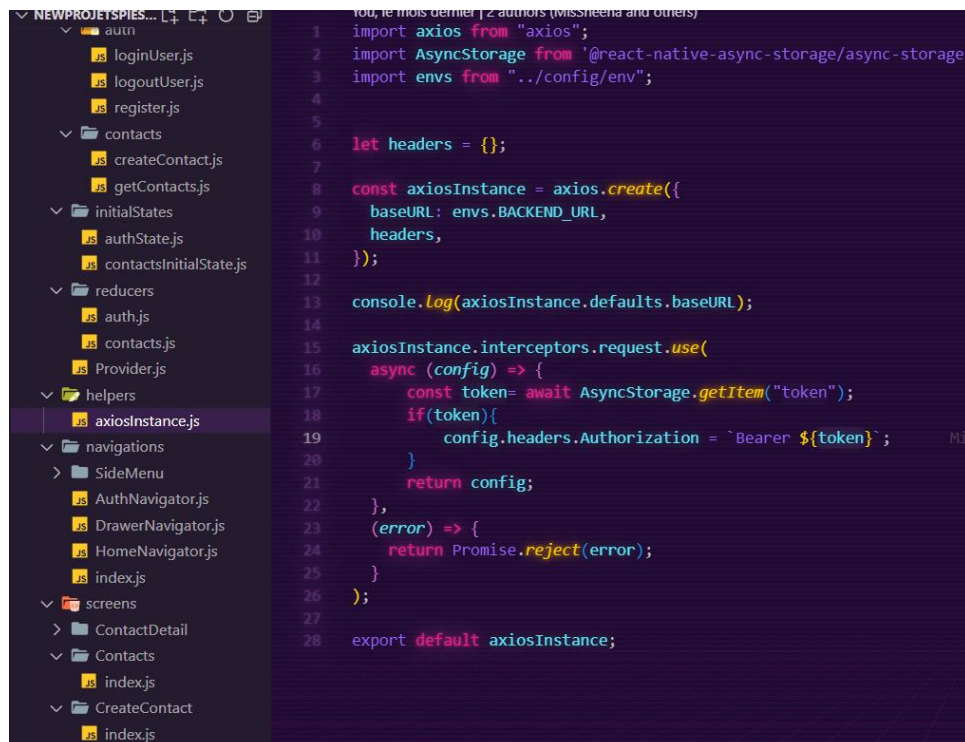
Ainsi, lors du dispatch, il y a juste à préciser les cas

Et l'application sait automatiquement les actions à effectuer

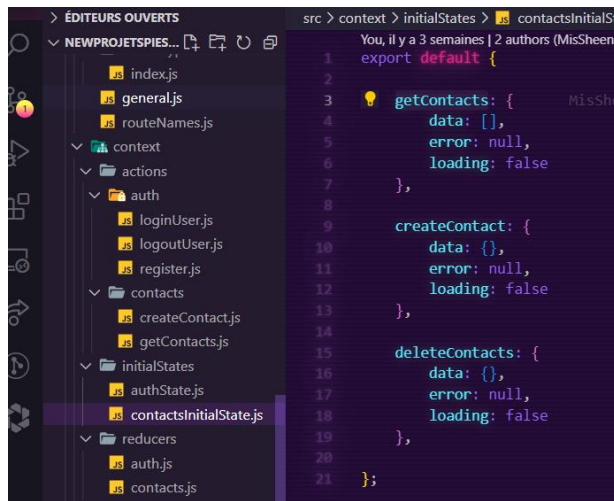
Car elles ont été paramétrées en amont dans le reducer.



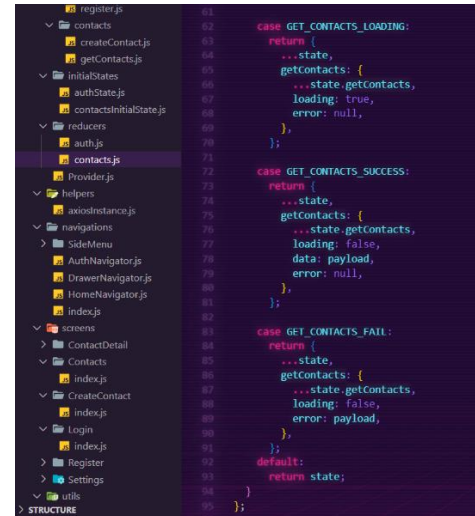
Construction de l'en tête axios



Pour gérer les contacts on aura créé des states initiaux



Pour pouvoir les utiliser dans le réducteur



Afficher la vue Mobile de la liste de client :

Test de l'application

Pour les tests E2E, la documentation suggère d'utiliser les testeurs [Detox](#) ou [Appium](#) . Ces runners nécessitent que l'application soit construite avant d'exécuter les tests.

Après avoir inspecté la liste des applications open source React Native sur [ReactNativeNews/React-Native-Apps](#), nous n'avons trouvé aucun test significatif dans la grande majorité d'entre elles !

Pour faciliter nos tests, nous allons utiliser à la fois Cypress et Jest . Nous pensons que ces outils se complètent et nous aideront à obtenir une bonne couverture de notre code. Nous utiliserons Cypress pour nos tests de bout en bout car nous l'avons trouvé assez convivial. Jest sera utilisé pour nos tests unitaires car nous avons vu combien de grandes entreprises l'utilisent avec beaucoup de succès.

Veille

Veille Sécurité

Il existe de nombreux outils permettant d'avoir une veille technologique en continu. Parmi eux, se trouvent les réseaux sociaux tel que Twitter, Facebook, Instagram et LinkedIn. Mais il existe également les sites web présentant des articles, comme GoogleActualité, Alvinet et OWASP.

Alvinet est un moteur de recherche qui permet de cibler grâce à des mots clés les sujets qui nous intéressent. Contrairement à GoogleActualité, qui quant à lui, est un service en ligne gratuit qui recueille des articles d'informations en provenance de sources sur le web.

Sur les réseaux sociaux j'ai décidé de m'abonner au compte de la CNIL et l'ANSSI Officiel, mais également à plusieurs comptes nommés « Le journal informatique » et « developpez.com » sur Twitter,

L'Open Web Application Security Project (OWASP) est une organisation internationale à but non lucratif dédiée à la sécurité des applications Web. L'un des principes de base de l'OWASP est que Tous ces documents sont disponibles gratuitement et facilement accessibles sur son site internet, ce qui permet à chacun d'améliorer la sécurité de ses applications web. Le matériel qu'ils fournissent comprend des documents, des outils, des vidéos et des forums.

Grace au top 10 des failles de cyber-sécurité, publié tous les ans par l'OWASP, nous avons pu préparer notre application à la faille SQL qui consiste à envoyer des injections SQL dans les formulaires. Nous avons pu sécuriser ce problème.

Veille des technos mobiles :

Kotlin est un langage de programmation orienté objet et fonctionnel, avec un typage statique qui peut être compilé pour la machine virtuelle Java et qui est aujourd'hui considéré comme le langage recommandé pour le développement d'application Android. Le langage s'apprend relativement rapidement (la syntaxe est assez familière) et la documentation dispose d'une très bonne série de contenu qui permet d'apprendre progressivement les bases du développement android.

Les plus

- Une très bonne documentation
- Interopérable avec Java et son écosystème
- Des patterns bien définis pour structurer le code
- Très bonne expérience de développement avec Android Studio

Les moins

- On ne cible qu'une seule plateforme
- Composer des vues peut s'avérer complexe (A voir avec jetpack compose)
- Il faut recompiler pour voir les changements
- Des erreurs parfois difficiles à déboguer (dans le cas d'une erreur dans le XML par exemple)
- Verbeux

Développement cross-platform avec Flutter

Flutter est une technologie qui permet de développer des application cross-platform avec un seul et même code. Il se repose sur le langage Dart, qui est un langage de programmation orienté objet avec un typage statique qui peut être utilisé avec une compilation just-in-time (JIT) pendant le développement et ahead-of-time (AOT) pour la production.

Pour le rendu, Flutter utilise son propre moteur et ne se repose pas sur les composants natifs.

Les plus

- Une interface unique quel que soit la plateforme
- Le rechargement à chaud
- De nombreux composants offerts par défaut
- Un fonctionnement simple (découpage en composant)

Les moins

- La syntaxe n'est pas très lisible pour déclarer les vues
- Le système de navigation n'est pas clair
- Les performances du moteur de rendu peuvent être inconsistantes et difficile à déboguer.

React native permet d'utiliser la librairie React pour développer des applications natives. Le code ne va pas être compilé en natif mais sera exécuté par un moteur JavaScript qui communiquera avec la couche native pour générer l'interface..

Les plus

- Facile à prendre en main si on connaît react
- Un écosystème bien fourni

Les moins

- Il faudra chercher des librairies tierces très rapidement
- Toujours pas en version 1.0
- Il faudra faire attention aux performances
- Certains composants n'ont pas la même apparence suivant les plateformes

Conclusion

Au-delà des retours sur les différentes approches l'important est ici de vous montrer comment se passe la veille et comment on peut rapidement explorer les spécificités de certaines technologies pour se faire une idée de leur fonctionnement. Ce travail de veille est important car il permet de comprendre les besoins auxquelles ces technologies répondent et quand leur utilisation est optimale.

- Kotlin, si on ne cible qu'Android et que l'on veut utiliser les éléments natifs.
- Flutter, si on veut une interface cross-platform identique et si on utilise Material Design.
- React native, si on connaît déjà React.

Remerciement

Je souhaite remercier l'entreprise Hackers-Corporation de m'avoir accueillie au sein de son équipe tout au long de ces années et d'avoir contribué à mon apprentissage du métier de Concepteur web. Associée à celle-ci, tous les jours, j'ai pu acquérir de nouvelles compétences clés telles que la gestion de projets en tant que chef de projet. Grâce à cette expérience, j'ai pu acquérir des automatismes mais également des connaissances nouvelles et essentielles au métier de Développeur Web. Merci à Monsieur DESCODT Christophe pour la patience dont il a fait preuve, mais aussi pour ses explications détaillées qui m'ont permises de reprendre les logiciels de l'entreprise sans problème. Ces outils me seront bénéfiques et me permettront d'être opérationnelle pour mes futures expériences professionnelles. Le fait de travailler en totale autonomie, sur des outils que je n'avais jamais utilisés n'a que confirmé ma passion pour le développement. Désormais, je suis définitivement convaincue de m'orienter dans le secteur qui me convient. Je souhaite également remercier l'école, qui m'a permis de continuer mes études au sein de leurs établissement mais également de m'avoir permis de rencontrer des personnes motivées, dont l'équipe de SPIES IMMO, avec lesquelles je pourrais mener à bien des projets. Avec toute ma reconnaissance, je vous prie à tous et à toutes, de bien vouloir recevoir les expressions de mes remerciements les plus sincères.