



Dossier de projet professionnel

2022

MARINA SEGOR

Réalisation de l'application web et mobile

Sommaire

Introduction.....	4
Présentation en anglais.....	4
Description du projet.....	4
La problématique à résoudre	4
La Solution que nous proposons.....	5
Conception générale.....	6
Exigences fonctionnelles	6
Exigences/contraintes du cahier des charges.....	6
Données des utilisateurs	7
Gestion de projet.....	8
Equipe.....	8
Organisation	8
Outils collaboratifs.....	8
Conception et Prototypage de l'Application.....	11
Panel Utilisateurs.....	11
Story Map	12
Wireframe	12
UI Kit & Logo	13
Maquettes	14
Conception BDD.....	15
Conception Back-end	19
Api Plateforme Symfony	19
Spécifications fonctionnelles	21
Réalisation Back-end.....	23
La sérialisation.....	Erreur ! Signet non défini.
Les opérations API Plateforme :	23
Connexion avec JwtToken.....	Erreur ! Signet non défini.
PostMan :	27
Conception front-end.....	28

Choix des technos	28
Pourquoi expo ?.....	29
Arborescence de l'application	30
Réalisation Front-end:	32
Formulaire de connexion.....	32
Get des contact avec axios	35
Test de l'application	39
Veille	39
Veille Sécurité	39
Veille sur les outils de développement	40
Veille des technos mobiles :	40
Jeu d'essai	Erreur ! Signet non défini.
Remerciement.....	Erreur ! Signet non défini.

Introduction

Présentation en anglais

Hello ! My name is Marina, and I'm 40, and I'm working at myCharlotte.
And what is myCharlotte ?

It's a website which only purposes is to help people who have cancer, to make them feel better on a daily basis

We have what we call partner patients, who fought against cancer once, and who now can guide others, listen to their struggles, and advise them about the different methods that can make them overcome their difficulties, like, practicing yoga, médiation, Pilates and so on...exercices.

They built a first version with no UI/UX, of the website during the pandemia, which was right on point, because people with diseases felt more lonely.

But now they have more and more patients to handle, they have decided to improve it.

And that's why they have recruiting a new team including myself.

But, I'm here today to talk to you about our project school that we called Spies Immo

Description du projet

La problématique à résoudre

Un agent immobilier doit gérer ses clients et leurs documents, ils doivent suivre leur 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.

La Solution que nous proposons

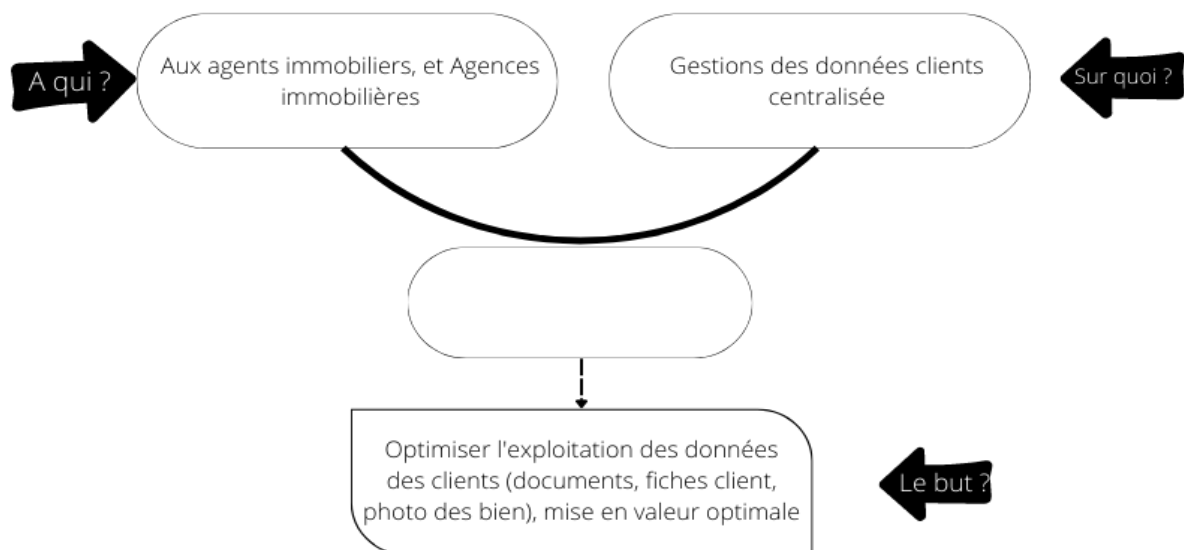
On propose une application mobile, un CRM de gestion clients, dans lequel un agent peut gérer tous ses dossiers clients. Où seront répertoriées toutes les informations nécessaires lors d'une location ou d'une vente d'un bien immobilier.

Pour faire gagner du temps aux agents, on propose une application où l'agent pourra créer et gérer ses clients. Grâce à l'application le suivi des documents sera simplifié et le rappel des documents manquant deviendra automatique.

Nous proposons un MVP, une application mobile qui permettra aux Agents de pouvoir créer et gérer leur client.

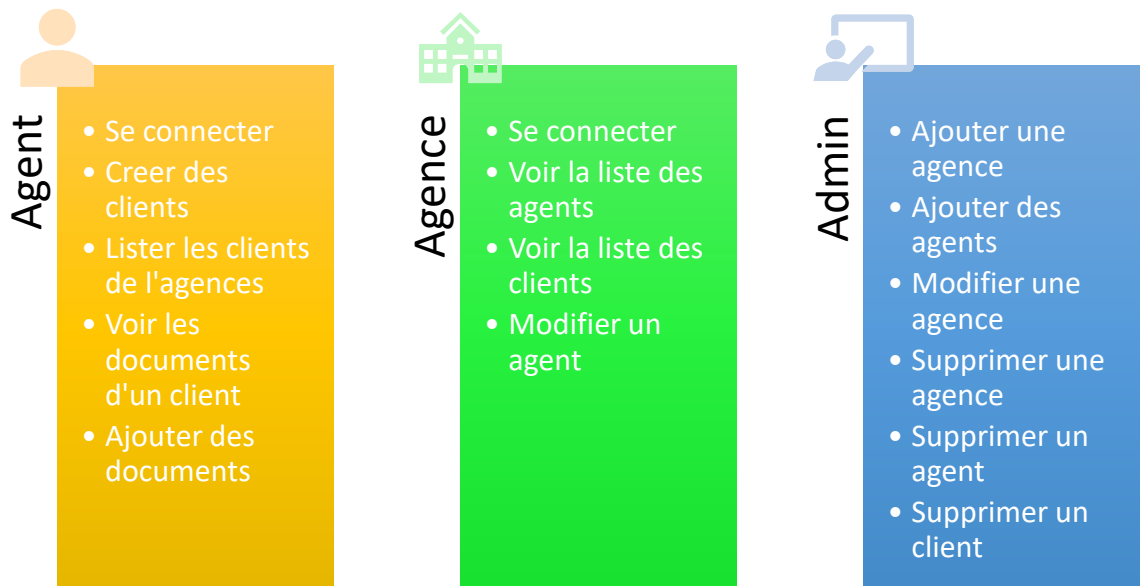
Une V2 dans laquelle un agent peut récupérer tous les documents d'un client dans un seul PDF et l'envoyer par mail à une tiers personne.

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*

Exigences/contraintes du cahier des charges

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	Le version mobile doit être codée avec React Native
EC6	La version Web ne doit être accessible qu'aux admin en react
EC7	Les test de la version web doivent être effectués avec Cypress
EC8	Les test en react native doivent être effectués avec React testing library & jest
EC9	Les test de L'API doivent être effectués avec PostMan

EC10	Les requêtes de L'API doivent être répertoriées 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 *En Réflexion niveau RGPD et ne concerne pas le MVP
EC16	Une fois désinscrit les données de l'agent ou l'agence seront conservées 6 mois
EC17	En cas de non-connexion pendant 1 an, le compte et les données de l'agence ou l'agent sont automatiquement supprimé

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.

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

Gestion de projet

Equipe

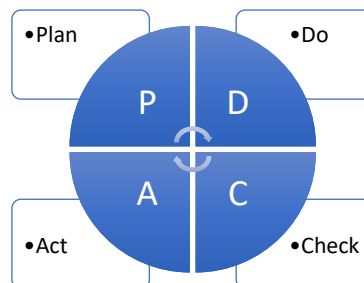
Marina Segor

Melina Hourcade

Maë Lodico

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 et les objectifs à atteindre

Outils collaboratifs



Discord

Nous avons utilisé discord pour la communication interne, car c'est un outil très permissif, auquel on peut ajouter beaucoup de plugin comme zapier, ou google agenda permettant une gestion du temps optimisée.

Drive

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

Whimsical

Pour la construction des wireframes

Figma

Pour la conception du logo, des maquettes, et du prototypage de l'interface utilisateur

VSCode

Pour le développement nous avons choisi de travailler avec Visual studio code, car il offre de nombreuses extensions qui facilitent grandement le développement ainsi que la collaboration avec la possibilité de faire du pair programming avec live share par exemple .

GIT

Git est un formidable outil de versionning, j'ai commencé à apprécier ce système de contrôle de version, après avoir lu un livre sur le sujet : "Git par la pratique" et qui explique que ce n'est pas seulement un outil de travail pour les développeurs, mais une philosophie d'organisation de travail.



Et c'est ce qui m'a donné envie de creuser la manière dont on l'utilisait au travail et de mettre en place dans notre projet, d'un commun accord les « conventionals commits »

Conventional Commits

A specification for adding human and machine readable meaning to commit messages

Il s'agit d'une convention qui définit un ensemble de règles afin d'écrire des messages de commits propres.

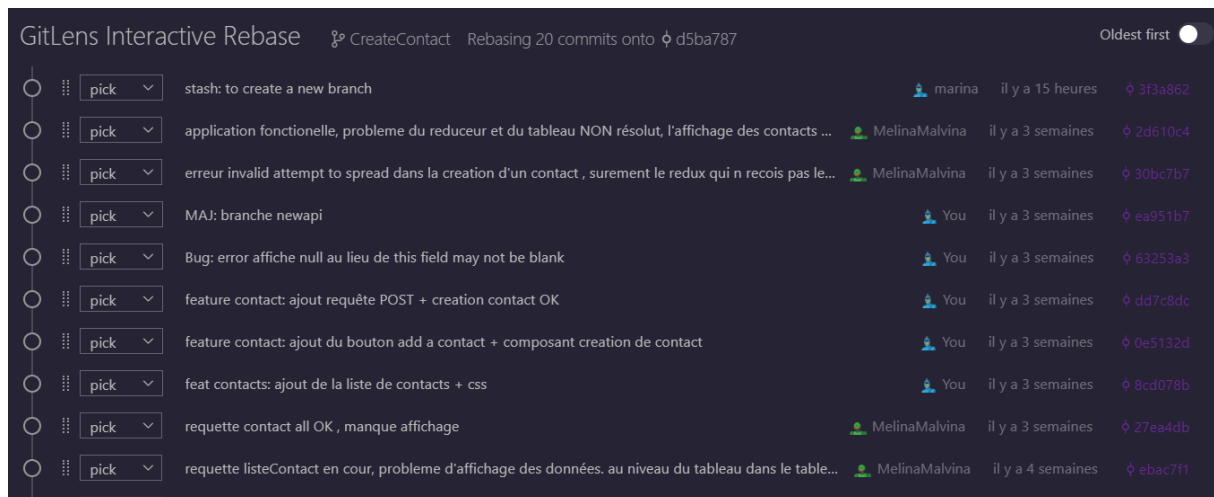
Exemple :

```
NewProjetSpiesImmo on  CreateContact []$  
→git commit -m "feat: create ContactComponent"
```

```
NewProjetSpiesImmo on  CreateContact [↑$]  
→git commit -m "fix: asyncstorage import"
```

Et ayant pas mal avancé dans le projet, il a fallu utiliser ce que GIT a de meilleur en terme de réécriture d'historique :

Le GIT rebase interactif



En effet, il offre plusieurs possibilités pour réorganiser ses commits, et avoir un historique propre, nous avons pris pour habitude de commit régulièrement, même si la feature n'est pas terminée, et on se retrouve rapidement avec un grand nombre de commits pas forcément pertinents à avoir dans notre repository.

Et en utilisant le squash, ça permet de fondre les commits les uns dans les autres, et ainsi de pouvoir les regrouper en feature, une fois qu'elle est terminée et fonctionnelle.

Autres possibilités du rebase interactif:

pick = utilise le commit

reword = utilise le commit, mais permet d'éditer le message du commit

edit = utilise le commit, change le message et le contenu du commit

squash = utilise le commit, mais se fond dans le commit précédent

fixup = Elle permet d'automatiquement réorganiser les commits dans le rebase interactif pour mettre les commits avec comme message fixup!

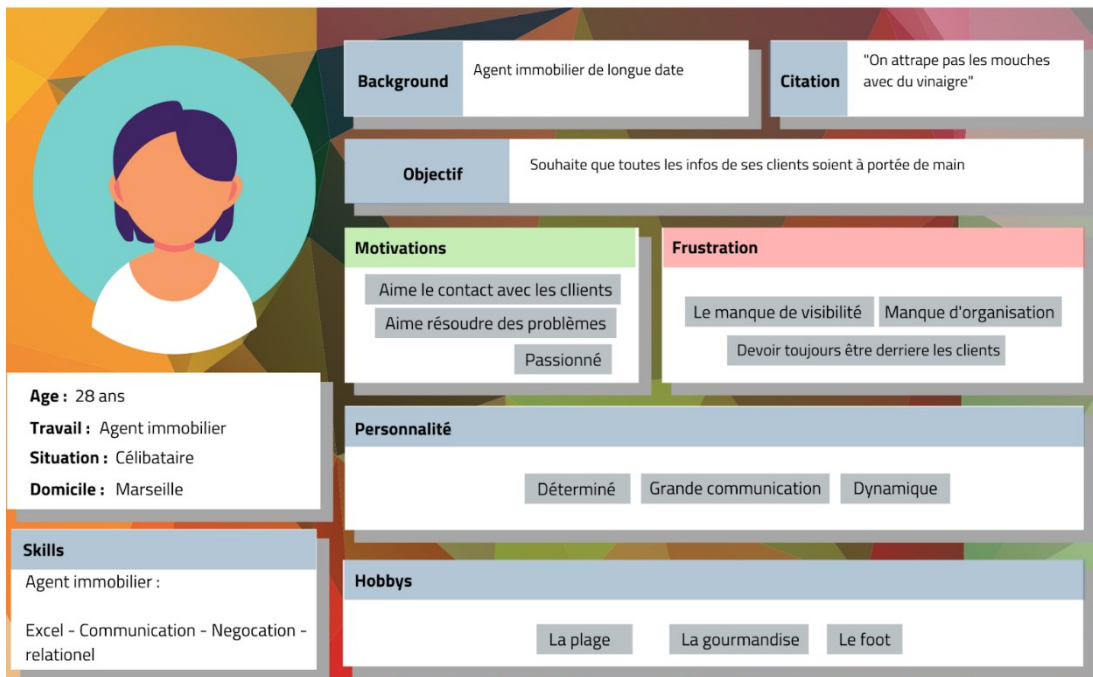
drop = supprime le commit

Conception et Prototypage de l'Application

Panel Utilisateurs

Nous avons constitué un panel de 5 utilisateurs potentiels en définissant leurs noms, âges, comportements, lieux d'habitation ainsi que leurs besoins/contraintes. L'objectif étant de faire tester l'application à un potentiel futur utilisateur.

Mélanie



A user profile card for Mélanie, featuring a circular avatar of a woman with dark hair. The card is divided into several sections with colored headers: Background (blue), Citation (blue), Objectif (blue), Motivations (green), Frustration (pink), Personnalité (blue), and Hobbys (blue). Each section contains specific details about the user's background, goals, motivations, frustrations, personality traits, and hobbies.

Background	Citation
Agent immobilier de longue date	"On attrape pas les mouches avec du vinaigre"

Objectif
Souhaite que toutes les infos de ses clients soient à portée de main

Motivations	Frustration
Aime le contact avec les clients Aime résoudre des problèmes Passionné	Le manque de visibilité Manque d'organisation Devoir toujours être derrière les clients

Personnalité
Déterminé Grande communication Dynamique

Hobbys
La plage La gourmandise Le foot

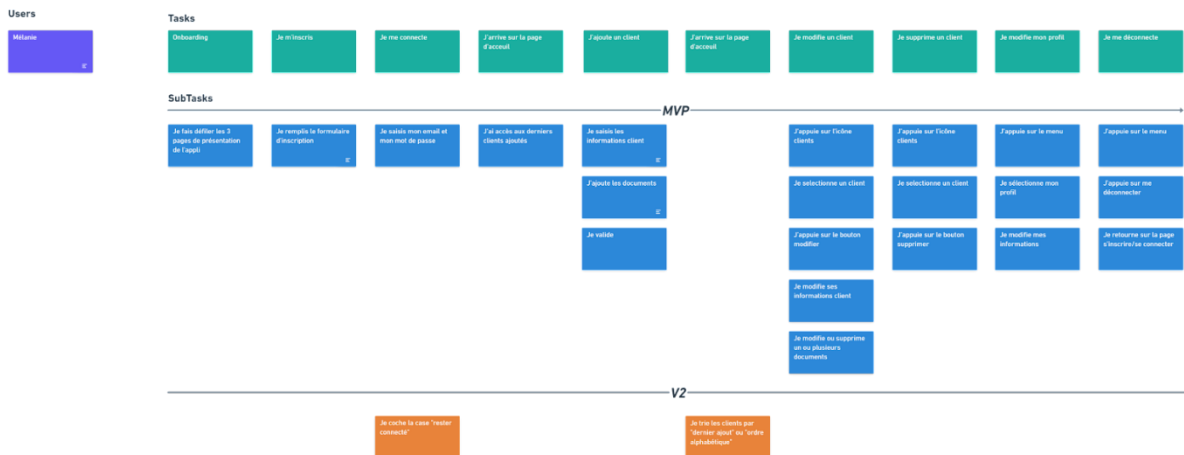
Skills
Agent immobilier : Excel - Communication - Negocation - relationel

Age : 28 ans	Travail : Agent immobilier	Situation : Célibataire	Domicile : Marseille
--------------	----------------------------	-------------------------	----------------------

Story Map

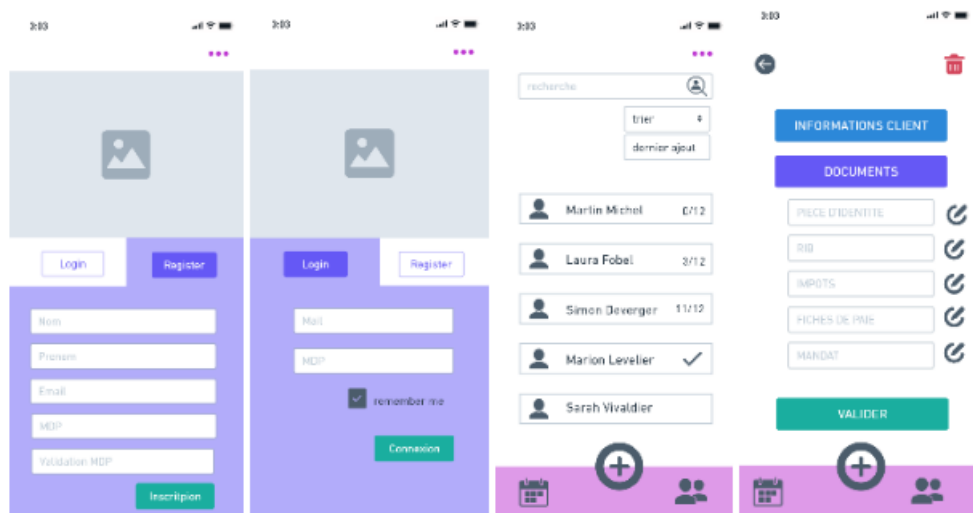
Afin de pouvoir évaluer le parcours utilisateur, en fonction des règles métiers, nous avons créé une story map, qui nous a servi, ensuite, pour la création de maquettes

User story map



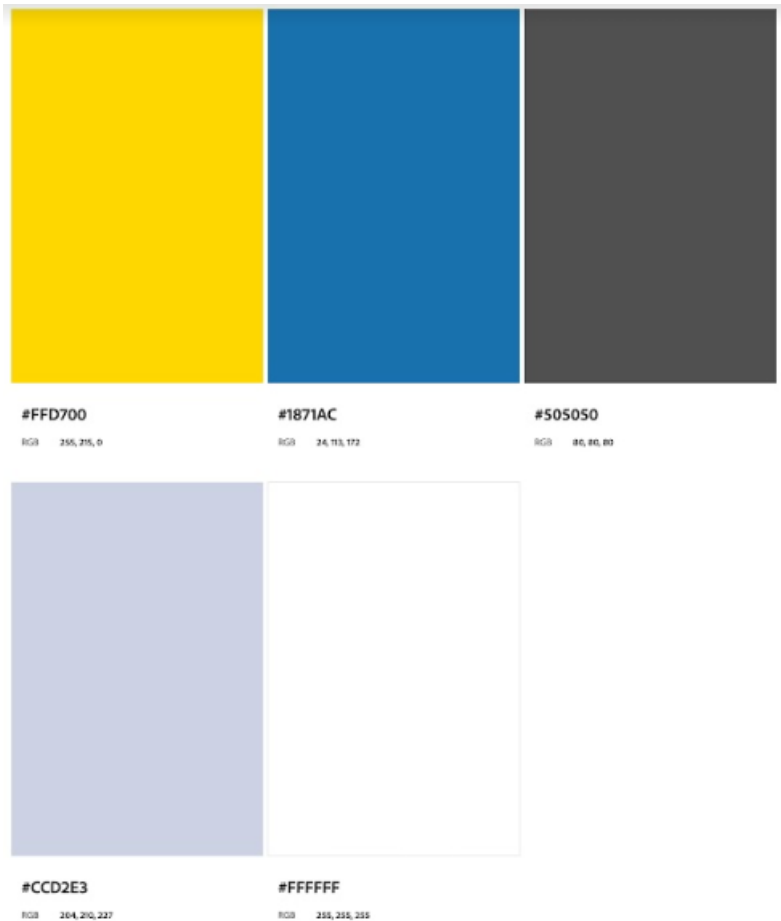
Wireframe

Avant de réaliser le prototype de notre application, nous avons créé des wireframes, pour la partie expérience utilisateur, voici quelques exemples d'écrans :



UI Kit & Logo

Pour notre charte graphique, nous avons choisi des couleurs sobres avec une touche de soleil (car on est quand même dans le sud !)

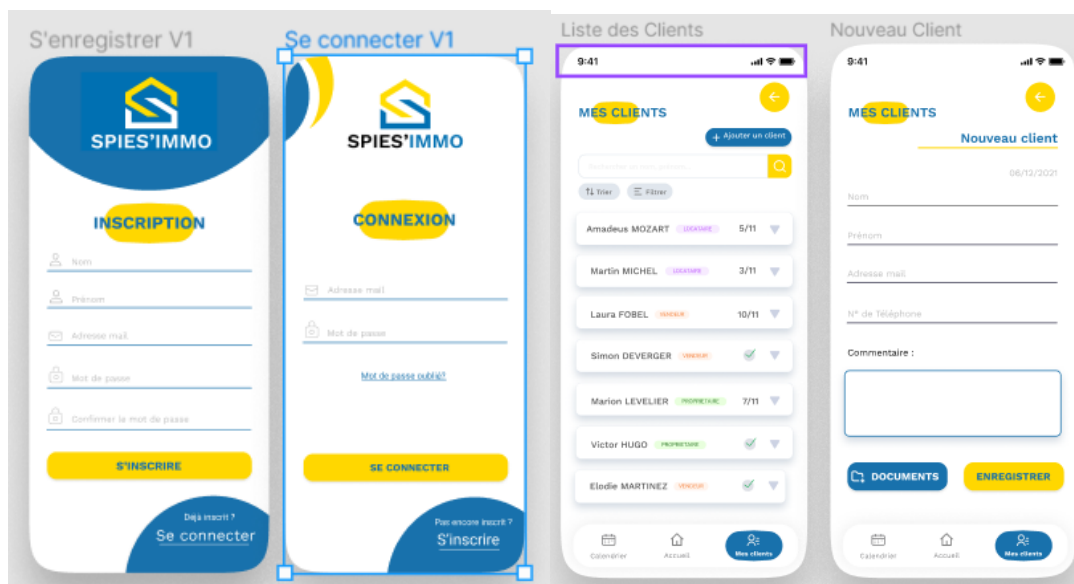


Avec des déclinaisons de Logo en fonction de la couleur du fond d'écran de l'application



Maquettes

Nous avons fait le choix de créer un onboarding, pour améliorer l'expérience utilisateur, et lui donner envie de découvrir l'application



Grace à ces maquettes, on a eu une meilleure vision des règles métiers attendues, et c'est ainsi qu'on a pu commencer à construire un MCD.

Pendant cette partie du développement les réunions nous nous sommes réunies plus régulièrement, afin de palier aux nombreuses questions techniques. Il y avait pas mal de question technique à prendre en compte concernant les choix pour l'application.

Cette méthodologie nous a permis une marge de manœuvre assez large concernant les décisions.

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.

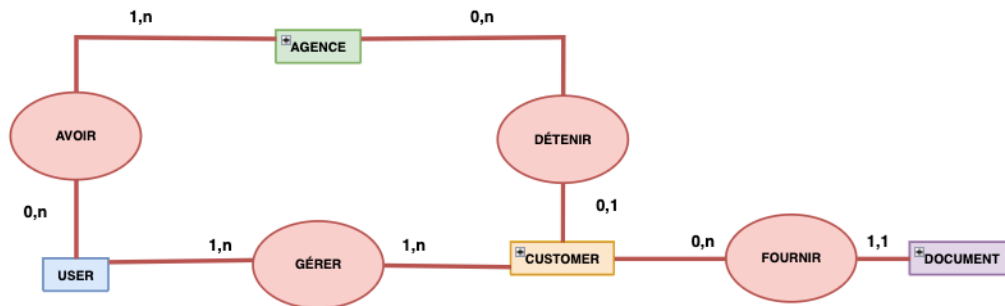
- AGENCE : **ID**, name, codeAgence.
- USER : **ID**, name, firstname, password, email, picture.
- CUSTOMER : **ID**, name, firstname, email, dateCreation, comment.
- DOCUMENT : **ID**, name, type, pdf.

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.

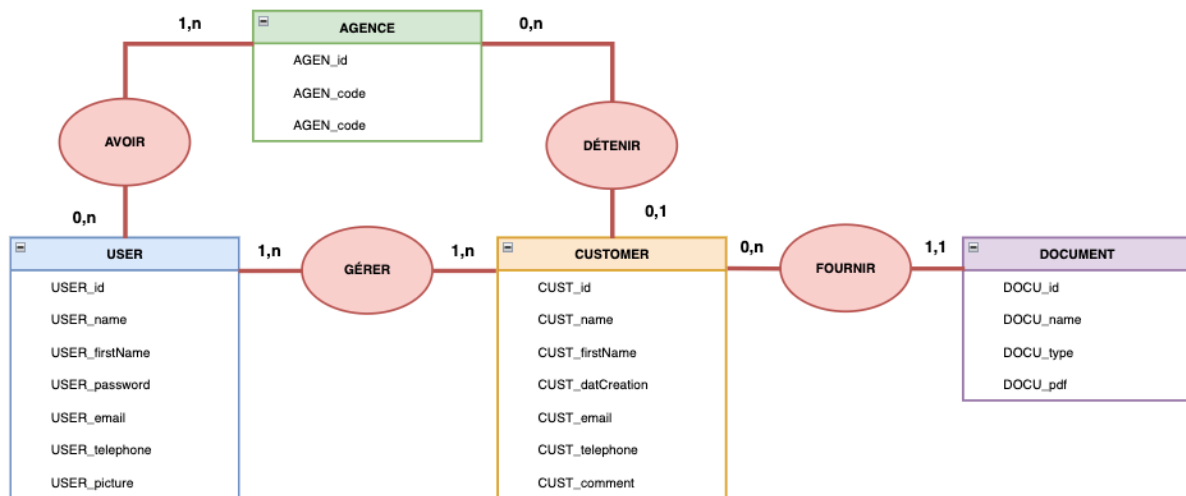
Étape 5 : Identification des cardinalités.



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.

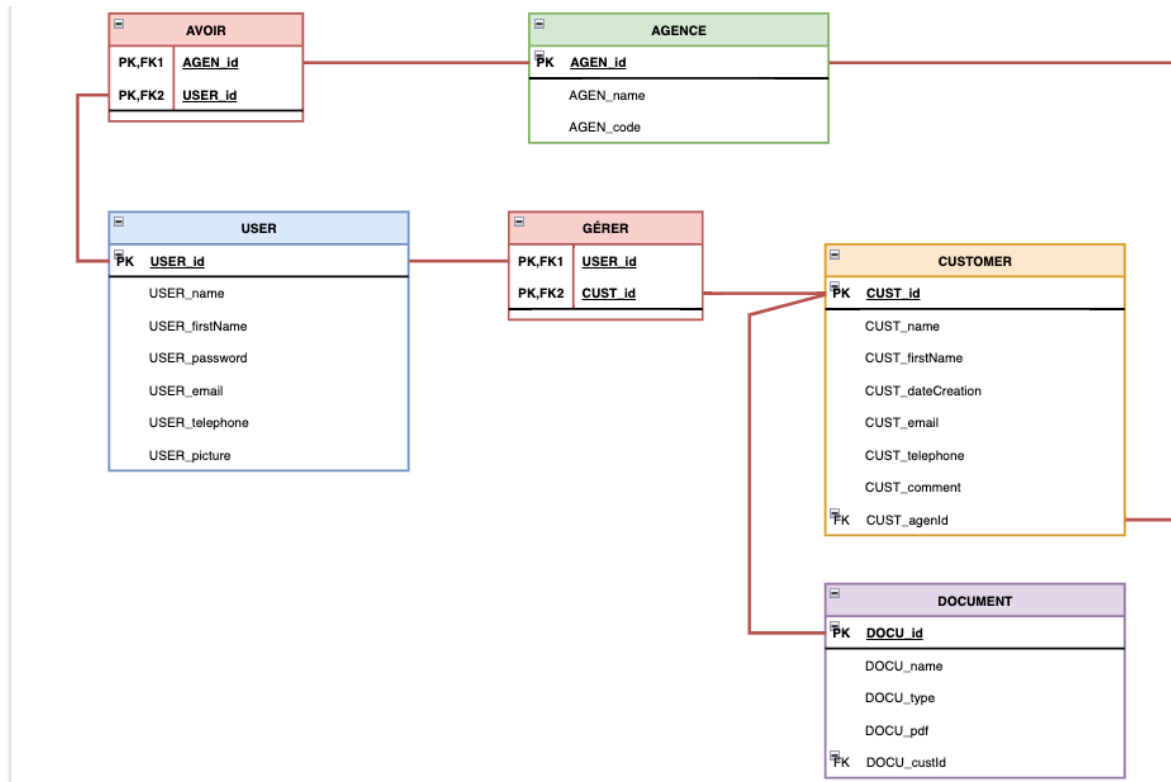


. PASSAGE DU MCD AU MRD -

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, (@Apiresources())

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

Spécifications fonctionnelles

Caractéristiques utilisateurs

Agent

Un nom
Un email
Un profil
Une liste de client

Agence

Un nom
Un Profil
Une liste d'agents

Actions utilisateurs

Agent

Create client
Read Client
Update Client
Delete Client
Create Document
Read Document
Delete Document
Rechercher Client de L'agence

Agence

Créer un agent
Modifier un agent
Supprimer un agent
Supprimer un client

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 agenc

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

Les opérations API Plateforme :

Création d'un modèle

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)
 * @ApiResource()
 */

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;
}

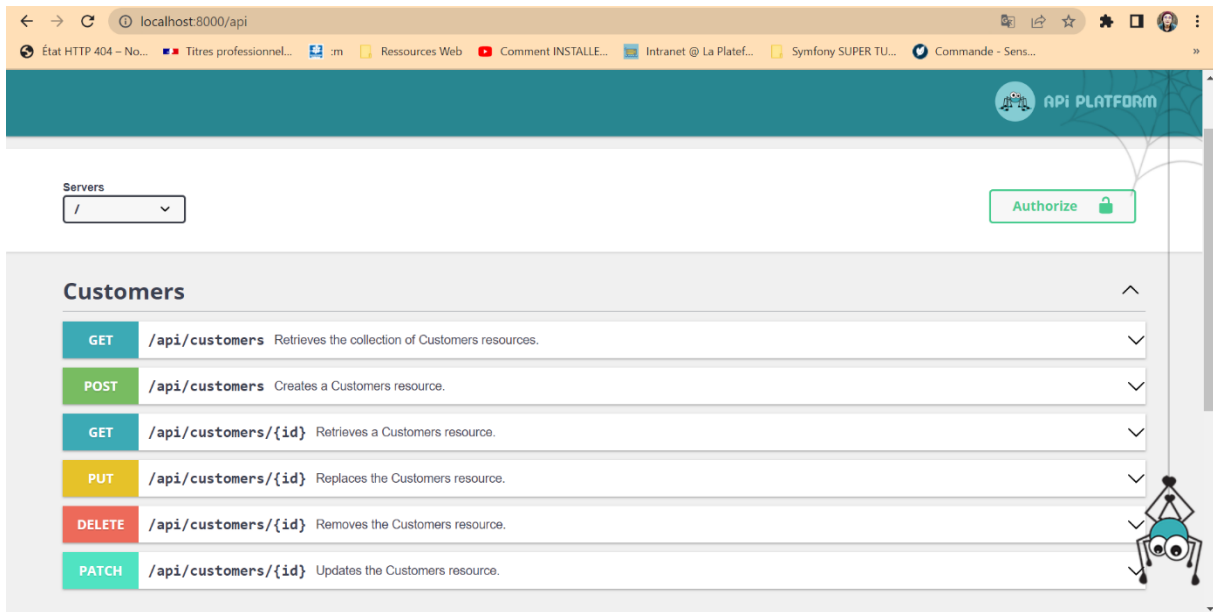
public function getName(): ?string
{
    return $this->name;
}

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.



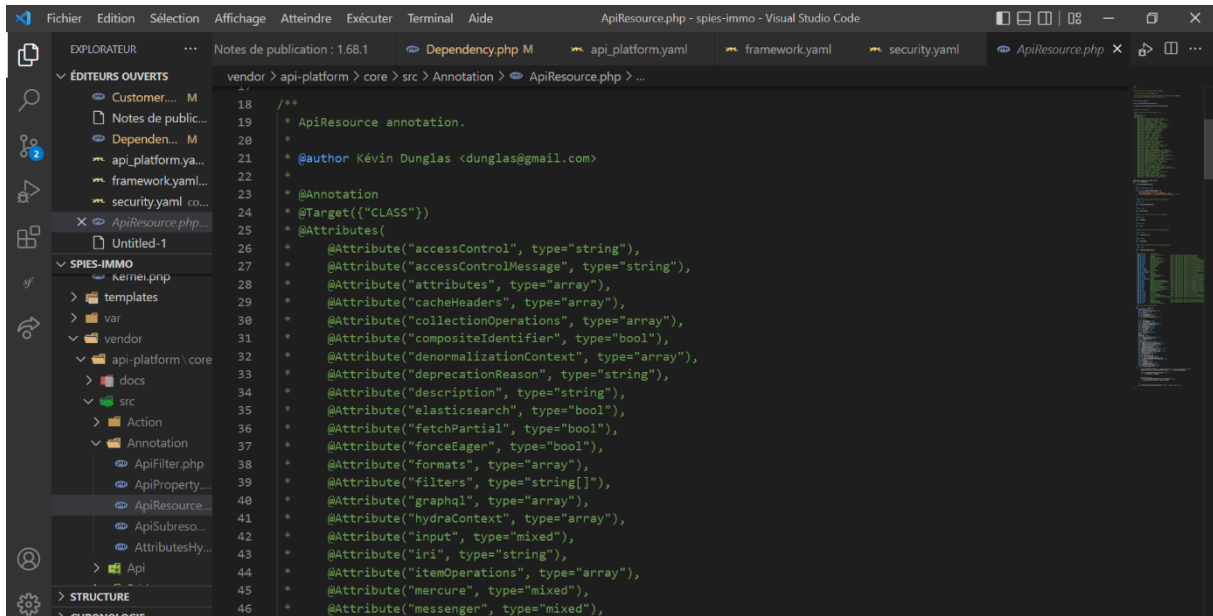
Modification de l'entité à l'aide d'attribut dans Api Plateforme

```
/**
 * @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.

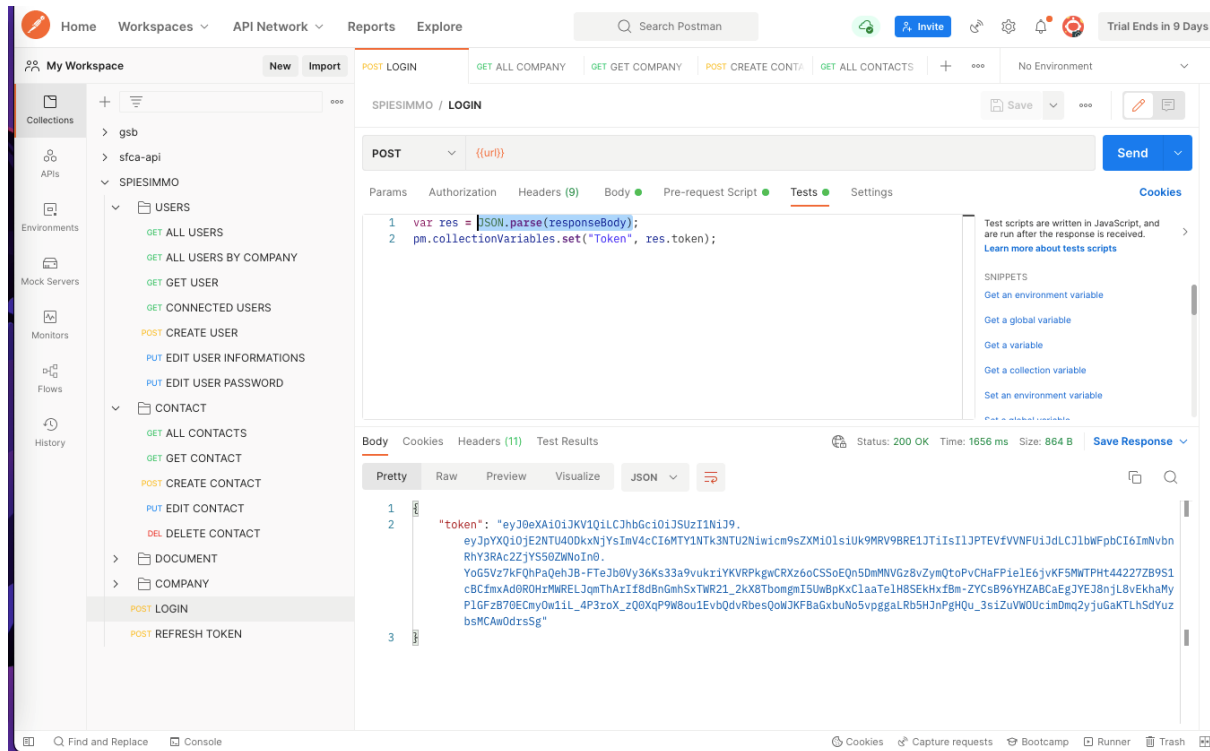
Notre documentation est basée sur une version antérieure à php8, nous sommes encore avec des annotations plutôt que des attributs, que la version 8 de php utilise.



```
18 /**
19  * ApiResource annotation.
20  *
21  * @author Kévin Dunglas <dunglas@gmail.com>
22  *
23  * @Annotation
24  * @Target({"CLASS"})
25  * @Attributes({
26  *     @Attribute("accessControl", type="string"),
27  *     @Attribute("accessControlMessage", type="string"),
28  *     @Attribute("attributes", type="array"),
29  *     @Attribute("cacheHeaders", type="array"),
30  *     @Attribute("collectionOperations", type="array"),
31  *     @Attribute("compositeIdentifier", type="bool"),
32  *     @Attribute("denormalizationContext", type="array"),
33  *     @Attribute("deprecationReason", type="string"),
34  *     @Attribute("description", type="string"),
35  *     @Attribute("elasticsearch", type="bool"),
36  *     @Attribute("fetchPartial", type="bool"),
37  *     @Attribute("forceEager", type="bool"),
38  *     @Attribute("formats", type="array"),
39  *     @Attribute("filters", type="string[]"),
40  *     @Attribute("graphql", type="array"),
41  *     @Attribute("hydraContext", type="array"),
42  *     @Attribute("input", type="mixed"),
43  *     @Attribute("iri", type="string"),
44  *     @Attribute("itemOperations", type="array"),
45  *     @Attribute("mercure", type="mixed"),
46  *     @Attribute("messenger", type="mixed"),
```

PostMan :

On écrit une collection postman avec des variables de connexion
on a ensuite fait des pre-scripts afin d'automatiser l' url à chaque envoi de requête on
récupère également automatiquement le token, **un script test de postman qui fait que on
n'est pas obligé de modifier le token plein à chaque fois**



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.

Conception front-end

Choix des technos

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.

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.

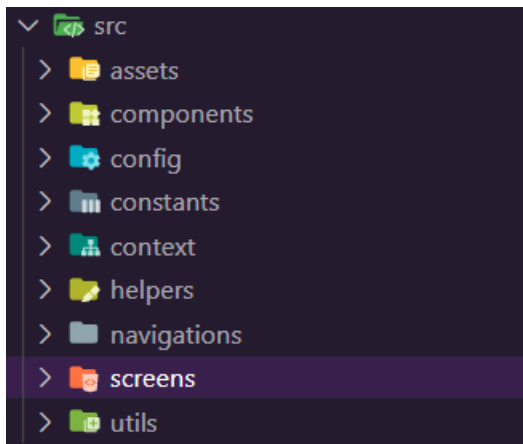
Ensuite, Expo est livré comme on l'a vu avec de nombreuses librairies React Native. C'est intéressant, certes... sauf que même si vous ne les utilisez pas toutes, votre application les chargera quand même.

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.

Arborescence de l'application



Faire une architecture dans une logique de déclinaison de composants:

Dossier assets :

- A. fonts
- B. images
- C. thème (dans lequel on définit les couleurs globales)

Ex. : Factorisation du style et centralisation des variables de style afin d'unifier les visuels.

Dossier components :

- A. common
 - Détails :** Création de la vue des composants clés de l'application (taille de l'écran, input, button, message)
- a. container (Qui englobe la vue globale de l'application)
- b. customButton
- c. input
- d. message
- B. Login
- C. SignUp
 - Détails :** Login et SignUp on retrouve le texte, les messages d'erreur et les interactions avec et entre les éléments de ces composants (inputs, buttons).

Dossier config :

On a créé le fichier env.js dans lequel on importe les deux variables du fichier .env de l'application, dans lequel on a importé l'URL de l'API du backend.

On déclare deux variables une de dev et une de prod, que l'on exporte à l'aide d'une ternaire pour rester en mode développement.

Et on a voulu utiliser le __DEV__ pour simuler un vrai environnement de développement/production.

Dossier constants:

Il centralise les variables des routes et des types d'action.

Dossier context :

A. actions/auth

Détails : Construction des requêtes AXIOS register et loginUser en méthode POST, on récupère les données des props (username du back + assignation du username du front et password), et affichage d'un message success ou fail selon le type de réponse reçue.

B. initialStates

Détails : Dans ces fichiers on définit l'objet et son état pour la connexion/inscription et pour les contacts.

C. reducer

Le provider permet de dispatcher ces états aux enfants.

Détails : On définit les cas d'état d'authentification et d'enregistrement d'un utilisateur. Et de création, suppression, modification d'un contact.

Dossier helpers:

fichier axiosInterceptor.

Détails : va permettre le futur stockage du token (en construction).

Dossier navigations:

Contient nos fichiers de navigation le fichier index.js vient vérifier si on est logués ou pas.,

Ex. : soit on affiche on affiche la page d'accueil avec le drawernavigator soit on affiche le auth navigator.

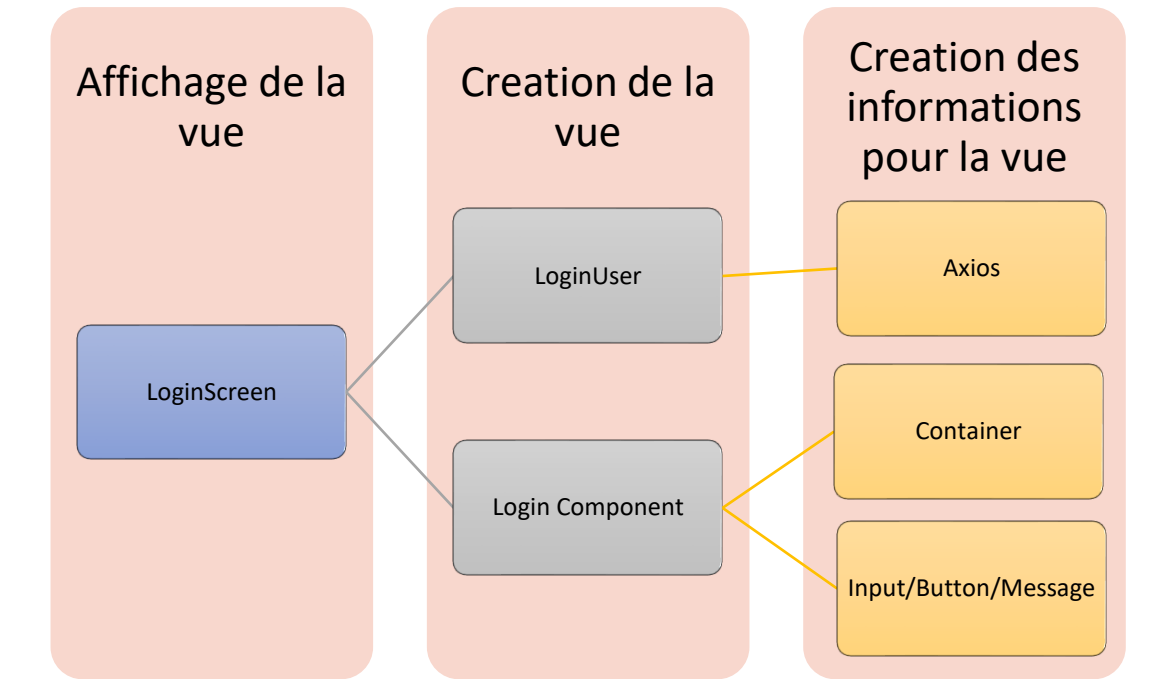
Détails : ensuite les différents composants authnavigator, drawernavigator, homenavigator, permettent de naviguer au sein de l'application

Dossier screens :

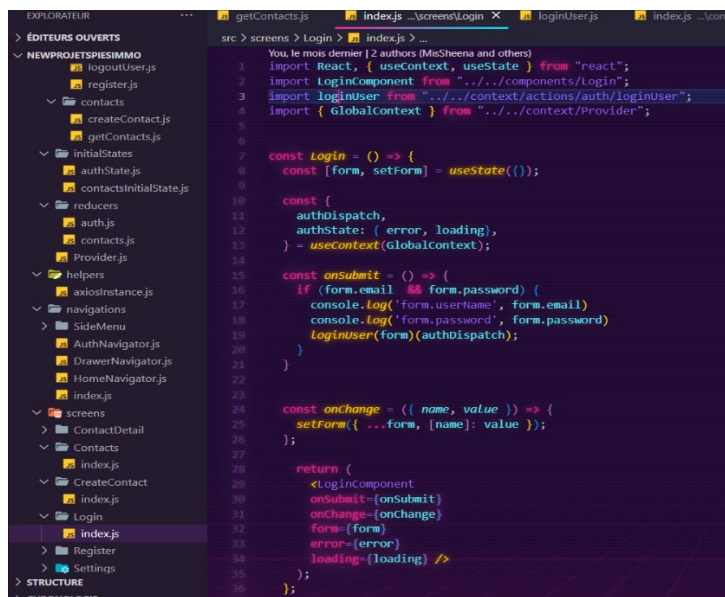
Ici on retrouve les composants contacts, login et register qui permettent d'afficher les règles de validation, dans ces composants on y appelle les composants qui se trouvent dans components/common pour l'affichage

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 }
```

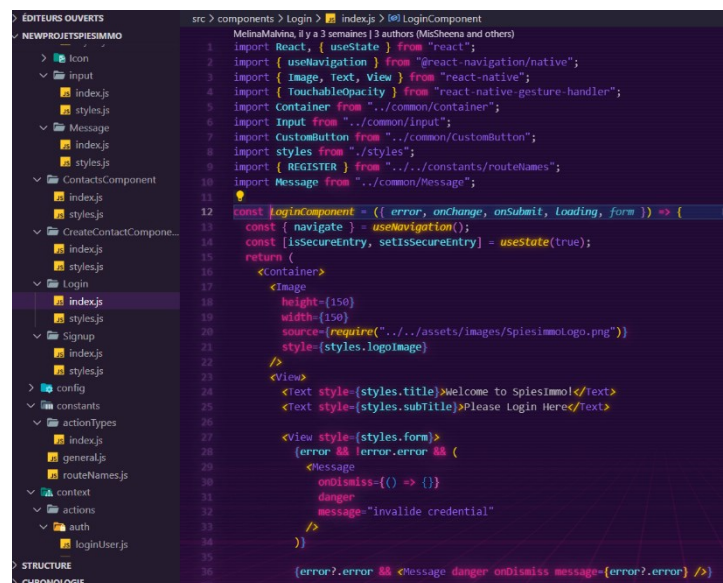
Il dépend de LoginComponent, comme son nom l'indique c'est ici ou sont créés les champs visuels qui serviront à construire la vue. On y importe plusieurs classes créées en amont, comme Input, ou Custom Button.

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 :

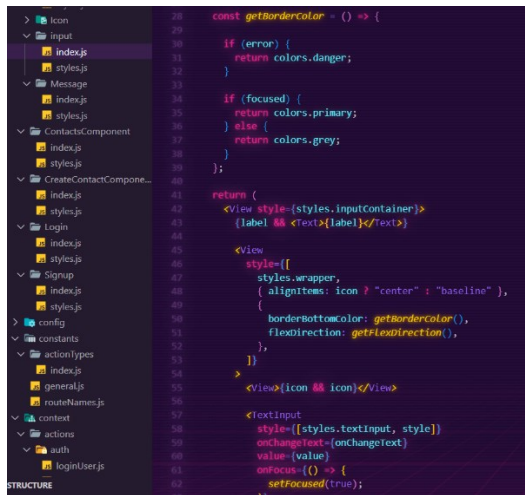
Comme mentionnée plus haut LoginComponent est constitué de plusieurs classes par exemple on vient définir la taille du container

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.



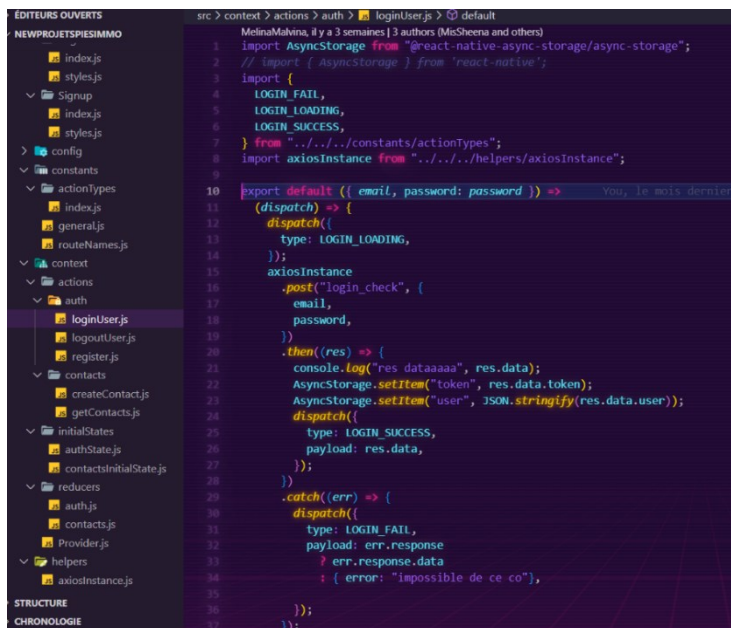
```
1 import React, { useState } from "react";
2 import { useNavigation } from "react-navigation/native";
3 import { Image, Text, View } from "react-native";
4 import { TouchableOpacity } from "react-native-gesture-handler";
5 import Container from "../../common/Container";
6 import Input from "../../common/Input";
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           onChange={onChange}
31           onDismiss={() => {}}
32           danger={danger}
33           message="invalid credential"
34         />
35       </View>
36       <CustomButton
37         error={error}
38         message={message}
39         onDismiss={onDismiss}
40         danger={danger}
41       />
42     </Container>
43   );
44 }
```

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



```
28 const getbordercolor = () => {
29
30   if (error) {
31     return colors.danger;
32   }
33
34   if (focused) {
35     return colors.primary;
36   } else {
37     return colors.grey;
38   }
39 };
40
41 return (
42   <View style={styles.inputContainer}>
43     {label} <Text>{label}</Text>
44
45     <View
46       style={
47         styles.wrapper,
48         { alignitems: icon ? "center" : "baseline" },
49       }
50       borderBottomColor={getbordercolor()},
51       flexDirection={getflexdirection()},
52     >
53       <View>{icon} </View>
54
55       <TextInput
56         style={([styles.textinput, style])}
57         onChangeText={onChangeText}
58         value={value}
59         onFocus={() => {
60           setfocused(true);
61         }}
62       />
63     </View>
64   </View>
65 );
```

⇒ LoginUser pour la requête :



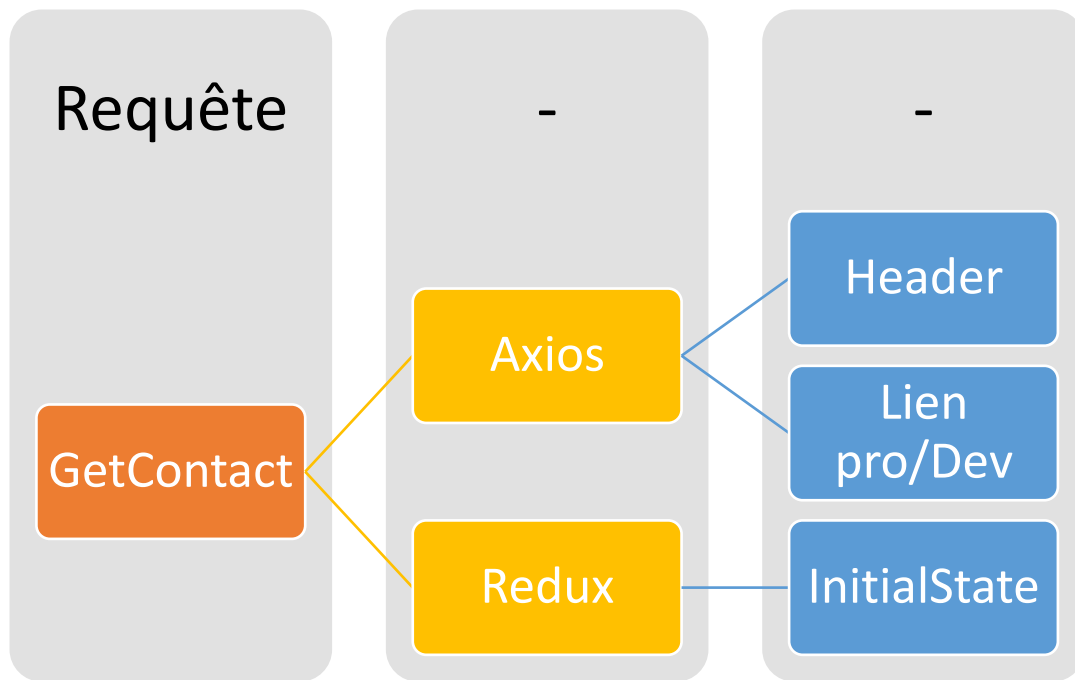
```
src > context > actions > auth > loginUser.js > default
MeinaMahina, il y a 3 semaines | 3 authors (MeSheena and others)
1 import AsyncStorage from "@react-native-async-storage/async-storage";
2 // import { AsyncStorage } from "react-native";
3 import {
4   LOGIN_FAIL,
5   LOGIN_LOADING,
6   LOGIN_SUCCESS,
7 } from "../../constants/actionTypes";
8 import axiosInstance from "../../helpers/axiosInstance";
9
10 export default ({ email, password }) => {
11   dispatch({
12     type: LOGIN_LOADING,
13   });
14   axiosInstance
15     .post("login_check", {
16       email,
17       password,
18     })
19     .then((res) => {
20       console.log("res dataaaaa", res.data);
21       AsyncStorage.setItem("token", res.data.token);
22       AsyncStorage.setItem("user", JSON.stringify(res.data.user));
23       dispatch({
24         type: LOGIN_SUCCESS,
25         payload: res.data,
26       });
27     })
28     .catch((err) => {
29       dispatch({
30         type: LOGIN_FAIL,
31         payload: err.response
32           ? err.response.data
33           : { error: "impossible de ce co" },
34       });
35     });
36   });
37 }
```

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 ?

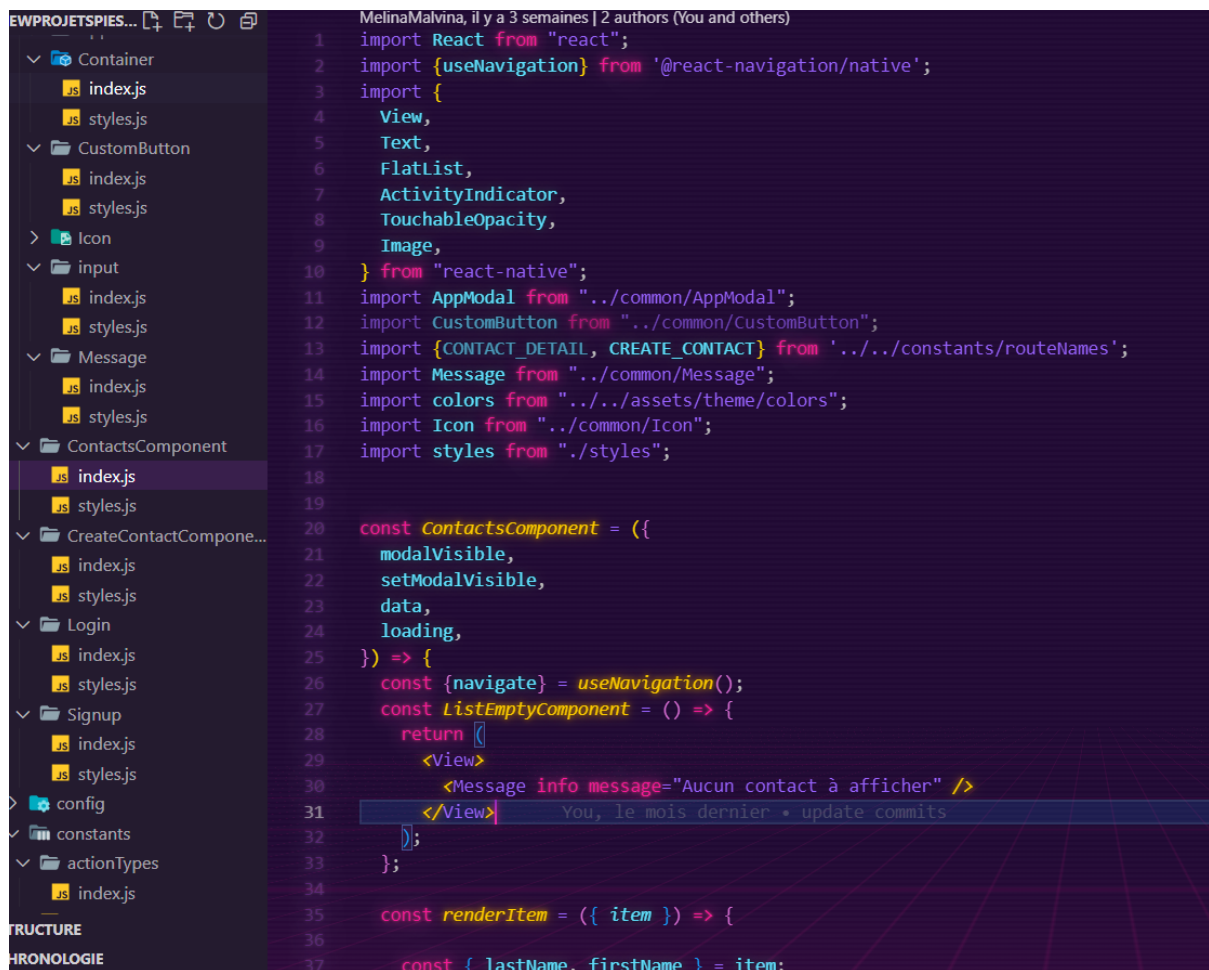


ScreenContacts

- 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       </View>
33     );
34   };
35
36   const renderItem = ({ item }) => {
37     const { lastName, firstName } = item;
```

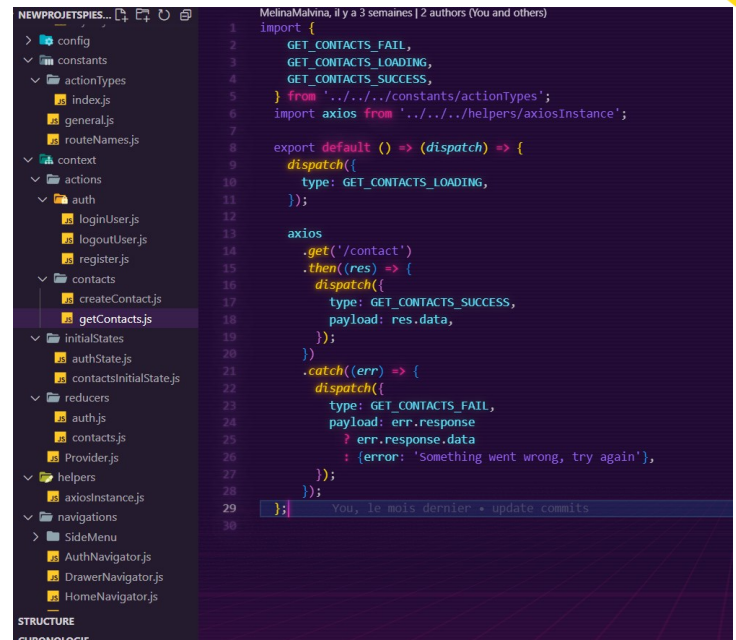
GetContacts

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

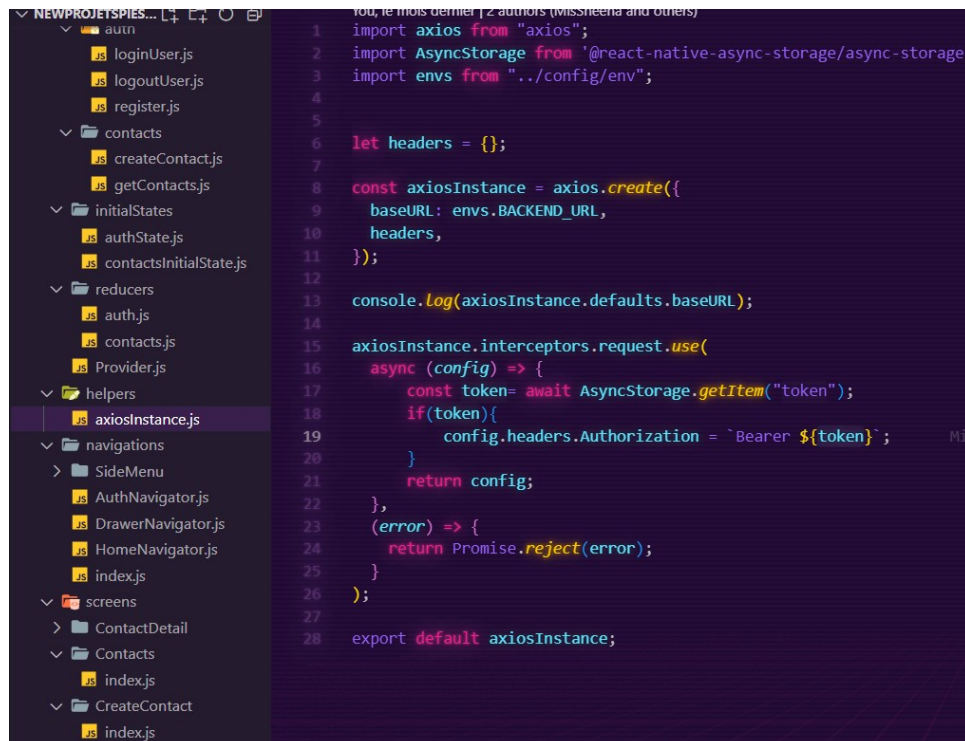
Et l'application sait automatiquement les actions à effectuer

Car elles ont été paramétrées en amont dans le reducer.



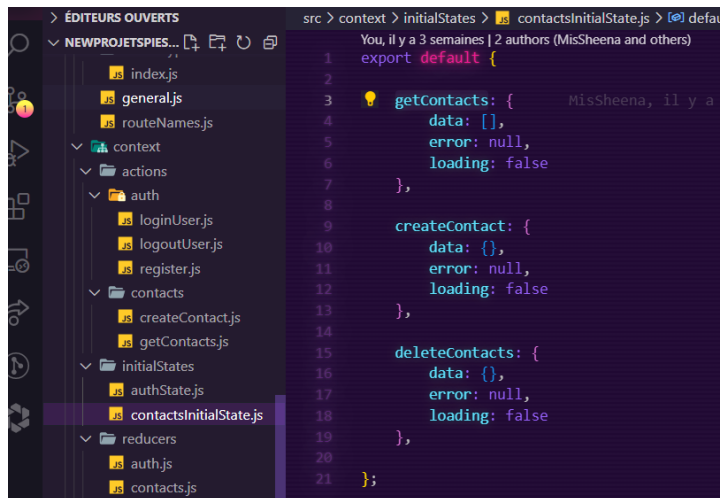
```
1 import {
2   GET_CONTACTS_FAIL,
3   GET_CONTACTS_LOADING,
4   GET_CONTACTS_SUCCESS,
5 } from '../constants/actionTypes';
6 import axios from '../helpers/axiosInstance';
7
8 export default () => (dispatch) => {
9   dispatch({
10     type: GET_CONTACTS_LOADING,
11   });
12
13   axios
14     .get('/contact')
15     .then((res) => {
16       dispatch({
17         type: GET_CONTACTS_SUCCESS,
18         payload: res.data,
19       });
20     })
21     .catch((err) => {
22       dispatch({
23         type: GET_CONTACTS_FAIL,
24         payload: err.response
25           ? err.response.data
26           : {error: 'Something went wrong, try again'},
27       });
28     });
29 });
```

Construction de l'en tête axios



```
1 import axios from "axios";
2 import AsyncStorage from '@react-native-async-storage/async-storage';
3 import envs from "../config/env";
4
5
6 let headers = {};
7
8 const axiosInstance = axios.create({
9   baseURL: envs.BACKEND_URL,
10   headers,
11 });
12
13 console.log(axiosInstance.defaults.baseURL);
14
15 axiosInstance.interceptors.request.use(
16   async (config) => {
17     const token = await AsyncStorage.getItem("token");
18     if(token){
19       config.headers.Authorization = `Bearer ${token}`;
20     }
21     return config;
22   },
23   (error) => {
24     return Promise.reject(error);
25   }
26 );
27
28 export default axiosInstance;
```

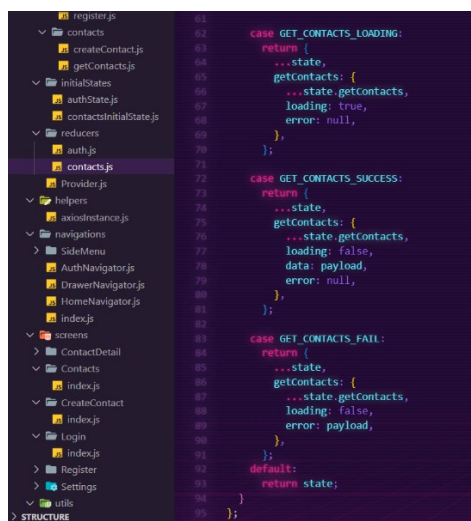
Pour gérer les contacts on aura créé des states initiaux



The screenshot shows the VS Code interface. On the left, the file explorer displays a project structure with folders like 'context', 'actions', 'auth', 'contacts', 'initialStates', and 'reducers'. The file 'contactsInitialState.js' is selected under the 'initialStates' folder. The main editor shows the content of this file, which defines three initial state objects: 'getContacts', 'createContact', and 'deleteContacts'. Each object has a 'data' array, an 'error' set to null, and a 'loading' set to false. A comment at the top of the file reads: 'You, il y a 3 semaines | 2 authors (MisSheena and others)'.

```
src > context > initialStates > contactsInitialState.js > default
1 You, il y a 3 semaines | 2 authors (MisSheena and others)
2 export default {
3   getContacts: {
4     data: [],
5     error: null,
6     loading: false
7   },
8   createContact: {
9     data: {},
10    error: null,
11    loading: false
12  },
13  deleteContacts: {
14    data: {},
15    error: null,
16    loading: false
17  },
18 };
19
20
21
```

Pour pouvoir les utiliser dans le réducteur



The screenshot shows the VS Code interface. On the left, the file explorer displays a project structure with folders like 'contacts', 'initialStates', 'reducers', 'auth.js', 'contacts.js', 'Provider.js', 'helpers', 'axiosInstance.js', 'navigations', 'SideMenu', 'AuthNavigator.js', 'DrawerNavigator.js', 'HomeNavigator.js', 'index.js', 'screens', 'ContactDetail', 'Contacts', 'CreateContact', 'index.js', 'Login', 'index.js', 'Register', 'Settings', and 'utils'. The file 'contacts.js' is selected under the 'reducers' folder. The main editor shows the content of this file, which defines three action types: 'GET_CONTACTS_LOADING', 'GET_CONTACTS_SUCCESS', and 'GET_CONTACTS_FAIL'. Each action type has a corresponding reducer function that updates the state. The 'default' case returns the state unchanged.

```
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
258
```


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 navons 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

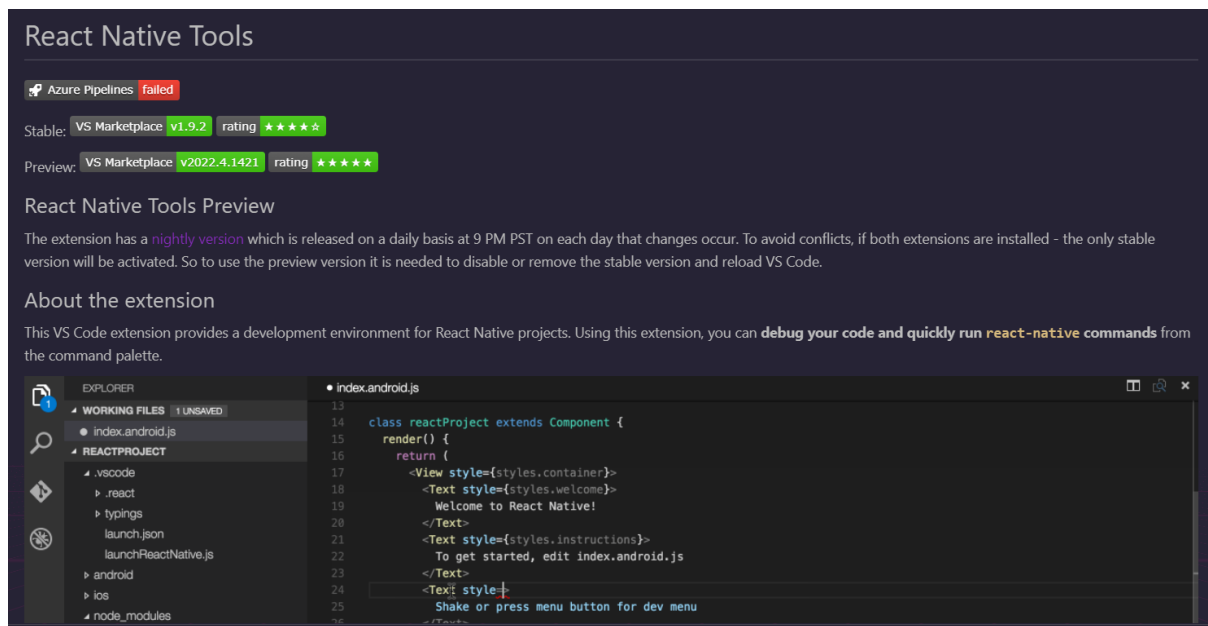
Il existe de nombreux outils permettant d'avoir une veille technologique en continu. Parmi eux, se trouvent les réseaux sociaux tel que Twitter, Facebook, Instagram et LinkedIn. Mais il existe également les sites web présentant des articles, comme GoogleActualité, Alvinet et OWASP.

Alvinet est un moteur de recherche qui permet de cibler grâce à des mot clés les sujets qui nous intéresse. Contrairement à GoogleActualité, qui quant à lui, est un service en ligne gratuit qui recueille des articles d'informations en provenant de source 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,

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 injonctions SQL dans les formulaires. Nous avons pu sécurisé ce problème.

Veille sur les outils de développement



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 tiers 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 quant 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.
- 