

Dossier de Projet Pro

Application Mobile Junior

Titre professionnel

Concepteur développeur d'application

École : La Plateforme

Entreprise d'accueil pour l'alternance : ART'CHI - 22/11/21 au 31/08/22

Prénom: Zoheir

Nom: MAATALLA

INTRODUCTION	2
RESUME DU PROJET	2
RESUME EN ANGLAIS DU PROJET « JUNIOR »	3
CAHIER DES CHARGES	5
DEFINITION DES BESOINS	9
LES OBJECTIFS DE L'APPLICATION	5
LES CIBLES	6
PERIMETRE DU PROJET	6
FONCTIONNALITES	6
CONCEPTION	11
VERSIONNING & WORKFLOW	11
USER STORY	13
Wireframe the junior Application	14
MAQUETTE & PROTOTYPE	16
MODELISATION DE LA BASE DE DONNEES MCD MLD MPD	17
DEPLOIMENT API	27
SPECIFICITES TECHNIQUES : BACKEND	31
SPÉCIFICITÉS TECHNIQUES : FRONT-END	34
LA SECURITE	40
Security(react-native)	
Symfony-Api-platefom)	
• Outils utilisés	42
• Conclusion	44

I. Liste des compétences du référentiel couvertes par le projet

Activité type 1 « Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité »	Maquetter une application	<input checked="" type="checkbox"/>
	Réaliser une interface utilisateur web statique et adaptable	<input checked="" type="checkbox"/>
	Développer une interface utilisateur web dynamique	<input checked="" type="checkbox"/>
	Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce	<input checked="" type="checkbox"/>
Activité type 2 « Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité »	Développer les composants d'accès aux données	<input checked="" type="checkbox"/>
	Développer la partie back-end d'une application web ou web mobile	<input checked="" type="checkbox"/>
	Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce	<input checked="" type="checkbox"/>

1. INTRODUCTION

1. 1. PRESENTATION PERSONNELLE

Je suis actuellement développeur web en alternance chez ART'CHI . Il y a maintenant 2 ans, j'ai entamé ma reconversion professionnelle pour devenir développeur web.

Après une première année de formation en développeur Web/Web Mobile, j'ai décidé de continuer sur une seconde année en alternance afin d'intégrer un environnement de professionnel et acquérir les compétences nécessaires à l'obtention du titre de Concepteur/Développeur d'applications.

Résumé du projet

Pendant nos périodes de formation, du temps a été consacré au développement d'une application mobile. Réalisée en commun avec Abdulrhaman Kamara, Toréa Patissier et Ahcene KADI des apprenants de mon cursus, nous avons nommé cette application « Junior ».

On est partie du constat, qu'il faut créer une plateforme qui permette la mise en relation des développeurs sans grande expérience du monde du travail, avec des entreprises recrutant ce genre de profil. Et cela sans avoir la concurrence de profil plus expérimenté afin de gagner du temps.

Junior, est une application qui permet de mettre en relation des développeurs junior et des entreprises. Le candidat doit créer un profil, après la création de son profil il aura accès aux offres publiées par les entreprises. Le candidat pourra effectuer une recherche par entreprises, ville ou spécialisation (ex : front-end etc..). Le candidat pourra postuler à une offre en remplissant un formulaire.

Les entreprises qui souhaitent s'y inscrire devront remplir un profil. Après

inscription, l'entreprise aura accès aux candidats possédant un compte. L'entreprise pourra effectuer des recherche par nom, ville ou spécialisation.

L'entreprise peut publier une offre d'emploi en renseignant un formulaire, celle-ci sera publié est à la vue des candidats uniquement.

RESUME EN ANGLAIS DU PROJET « JUNIOR »

I would like to introduce myself, my name is Zoheir MAATALLA, I am a second year student at the Platform, based in Marseille, in the second year, we have learned to develop apps, but we learned different kinds of frameworks and technologies, but we mostly focused on how to learn and build apps for Android and IOS.

As we know there are various mobile applications that may perform the same function as Junior. As developers it is always good to create something that answers the need of the client and the world at large. Junior App came just as what we thought as the little missing piece in job application. It's always heard for a junior developer to find their first job (stage, Alternance, CDD, CDI) in order to gain and improve the skills in web and application development. We this the idea of creation and application that serve the need of junior developer in helping us find our first experience in a company or a project. The aim is to provide job with all the junior developers within 0 to 3 years of professional experience

The aim of Junior is to help reduce the gap and doubt between junior developer that just want to find their first experience to put their skill to test and improve as better developer and Company Recruiters, and Start-ups. Junior is bringing you a meeting place for both the companies and the junior developers to have a direct contact. As the junior developer may have access to job opportunity that link to their level of skill and contact the company direct. Company that have posted a job offer can also view a junior developer profile and can contact him/her direct from the application.

COMPETENCES COUVERTES PAR LE PROJET

- Activité type 1 :

« Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité »

- Maquetter une application

- Développer des composants d'accès aux données

- Activité type 2

« Concevoir et développer la persistance des données en intégrant les recommandations de sécurité »

- Concevoir une base de données

- Mettre en place une base de données

- Développer des composants dans le langage d'une base de données

- Activité type 3 :

« Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité »

- Collaborer à la gestion d'un projet informatique et l'organisation de

- l'environnement de développement.

- Concevoir une application

- Développer des composants métier

- Construire une application organisée en couches

- Développer une application mobile

- Préparer et exécuter les plans de tests d'une application

- Préparer et exécuter le déploiement d'une application

Cahier des charges du projet

2.1. DEFINITION DES BESOINS

Il existe de nombreuses applications concernant la recherche d'emploi, cependant la volonté de l'application mobile « Junior » est de mettre en avant les profils ayant le plus de difficultés à trouver des postes, à savoir, les juniors.

Ainsi, en axant notre démarche en ce sens, nous permettons aux entreprises de bénéficier de jeunes talents, mais aussi au fait de les introduire dans le marché du travail.

2.1.1. LES OBJECTIFS DE L'APPLICATION

Les objectifs que l'on s'est fixé pour notre application sont les suivants :

- Faciliter l'insertion des jeunes dans le marché du travail
- Attirer de nouveaux prospects
- Obtenir des partenariats avec des entreprises
- Gagner en notoriété
- Développer notre communauté
- Augmenter notre visibilité

2.1.2. LES CIBLES

L'application sera ouverte à toute personne majeure, en demande d'emploi. La cible comprend toute personne qui cherche à et souhaitant rechercher une entreprise qui potentiellement recrute des Développeur Web sans expérience professionnelle.

L'application sera ouverte également aux entreprises qui ont un besoin de recruté des profils juniors.

2.1.3. PERIMETRE DU PROJET

Etant une application mobile, le projet ne sera disponible que pour les usagers de smartphone Android et iOS.

Junior est une application mobile native multiplateforme. Elle est disponible aux utilisateurs détenteurs d'un smartphone sous Android ou iOS et décliné en Français et en Anglais pour toucher un plus large public.

Celle-ci pourra comprendre l'utilisation de fonctionnalités natives des appareils mobiles, telles que l'appareil photo ou la géolocalisation.

Les données personnelles des utilisateurs seront stockées dans une base de données.

Dans sa première version, l'application n'offre pas la possibilité de partager ses données d'un utilisateur à un autre par n'importe quel système de partage.

2.1.4. FONCTIONNALITES

Application mobile :

Les écrans :

- L'utilisateur devra à son inscription choisir entre un statut de candidat « Junior » ou d'entreprise « Company »
- L'écran d'accueil : Présentation de l'application avec un splash screen, un bouton « Get Started » permettra à l'utilisateur d'être redirigé sur la page login.
- L'écran de choix avec deux boutons, un bouton « company » qui redirige l'utilisateur sur un formulaire dédié aux entreprises, un bouton « junior » qui redirige l'utilisateur sur un formulaire dédié aux candidats.
- L'écran inscription : Deux écrans d'inscription avec formulaire seront proposés,
 - Un écran pour le statut « Company » avec trois champs: « name » qui servira à renseigner le nom de l'entreprise, « email » qui servira d'username , « password » qui servira à créer un mot de passe et « confirm password » qui servira à confirmer le mot de passe renseigné juste avant. Un bouton « Sign up » redirigera l'utilisateur sur l'écran de son profil.
 - Un écran pour le statut « Junior » avec cinq champs : un champ « firstname » qui servira à renseigner le prénom, un champ « lastname » qui servira à renseigner le nom, un champ « email » qui servira à renseigner l'username du candidat, un champ « password » qui servira à créer un mot de passe et un champ « confirme password » qui servira à confirmer le mot de passe renseigné juste avant.

- L'écran connexion : Un formulaire de connexion avec deux champs: un champ « email » ou l'utilisateur devra entré son email renseigné a l'inscription et un champ « password » qui garantira la sécurité de son compte, un bouton « Sign in » qui redirige l'utilisateur sur le fil des offres d'emploi ou le fil des candidats en fonction de son statut a l'inscription « Junior » ou « Company ». Un bouton « sign up » redirige l'utilisateur sur la page d'inscription s'il n'est pas encore inscrit.
- L'écran profil : permettra a l'utilisateur de complété et/ou modifié ses données. Deux écrans seront proposé en fonction du statut de l'utilisateur a savoir « Junior » ou « Company »
- Un écran pour le statut « Junior », un formulaire avec neuf champs : un champ « avatar » permettra a l'utilisateur de mettre une photo de profil, un champ « firstname » ou le prénom de l'utilisateur entré a l'inscription sera visible est modifiable, un champ « lastname » ou le nom de l'utilisateur entré a l'inscription sera visible est modifiable, un champ « email » ou le mail entré a l'inscription sera visible mais ne pourra pas être modifié, un champ « phone » proposera a l'utilisateur de noté son numéro de téléphone, un champ « city » ou l'utilisateur devra renseigné sa ville de domiciliation, une liste « profession » déroulante ou l'utilisateur pourra sélectionné une des proposition suivant : Dev full-stack, Dev front-end, Dev back-end, une liste « experience » déroulante ou l'utilisateur pourra sélectionné une des proposition suivant : 1 an, 2 ans, 3 ans, une liste « Diploma » déroulante ou l'utilisateur pourra

sélectionné une des proposition suivant :BAC, BAC +1, BAC +2, BAC +3, BAC +4, BAC +5 et un champ « description » proposera a l'utilisateur de se décrire, un bouton « send » permettra de validé le formulaire et de redirigé l'utilisateur sur le fil des offres.

- Un écran pour le statut « Company », un formulaire avec 6 champs : un champ « logo » permettra l'utilisateur d'assigné un logo a son profil, un champ « name » ou le nom de l'entreprise entré a l'inscription sera visible est modifiable, un champ « address » ou l'utilisateur pourra entré l'adresse de l'entreprise, un champ « email » ou le mail entré a l'inscription sera visible mais ne pourra pas être modifié, un champ « city » ou l'utilisateur devra renseigné la ville de domiciliation de son entreprise et un champ « description » proposera a l'utilisateur de se décrire, un bouton « send » permettra de validé le formulaire et de redirigé l'utilisateur sur le fil des candidats.
- L'écran fil d'actualité pour le statut « Company », une liste de tout les candidat inscrit, une barre de recherche permettra de filtré les résultats
- L'écran fil d'actualité pour le statut « Junior », une liste de toutes les entreprises inscrise, une barre de recherche permettra de filtré les résultats
- L'écran de création des offres:
 - Il ne sera accessible qu'aux utilisateurs possédant le statut « Company », un formulaire... a d'écrire

- L'écran détail :
- Si l'utilisateur a le statut « Junior » il pourra affiché les détails des offres
- Si l'utilisateur a le statut « Company » il pourra affiché le profil des candidats

2.1.5. CONTRAINTES TECHNIQUES

L'application devant être compatible de manière native avec les systèmes d'exploitation mobile iOS et Android, plusieurs choix techniques ont été fait pour optimiser notre productivité.

Coté client, nous avons opté pour l'utilisation de la librairie React Native. Celle-ci permet de créer des applications mobiles native multiplateformes dans un langage unifié sans avoir à apprendre les langages respectifs de chaque système, elle offre un gain de temps et de productivité non négligeable.

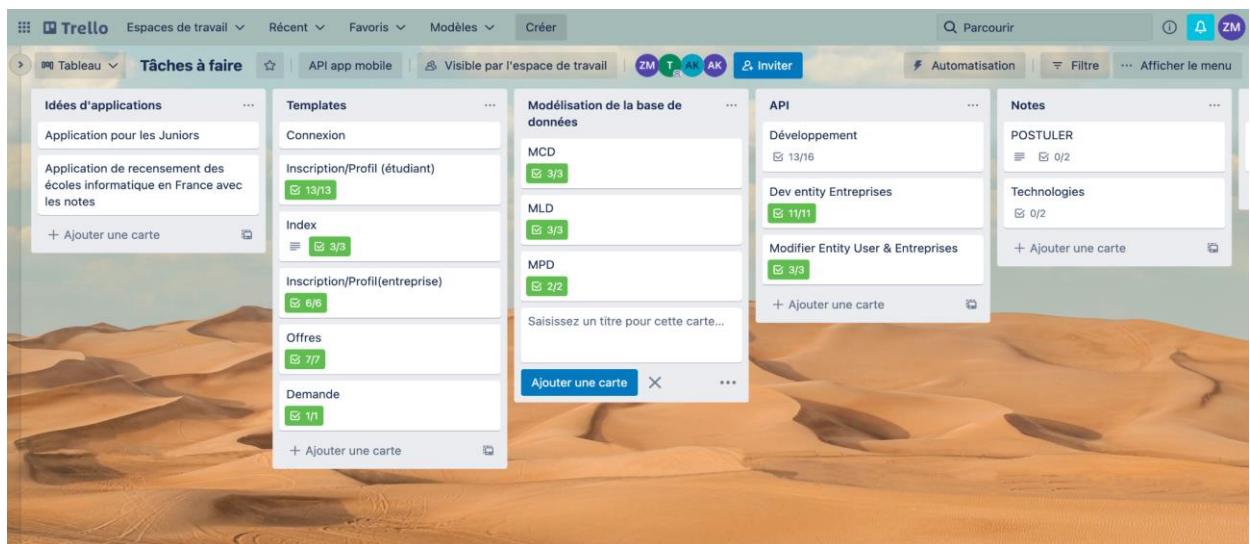
Coté Back-en notre choix c'est porté sur une base de données MySQL et Api-Platform pour l'API. La base de données et l'API qui permettent les requêtes HTTP devront être sécurisées. Les mots de passe de l'application devront être chiffrés en base de données et tout utilisateur pourra récupérer ou voir supprimer ses données à tout moment. Le code doit être scalable dans le temps.

3. CONCEPTION

Afin de mieux appréhender nos étapes de conception et d'intégration, nous avons fait le choix d'utiliser une approche AGILE. Étant quatre sur le projet et n'ayant pas de client externe, celle-ci nous a paru idéale pour optimiser au mieux notre flux de travail.

Pour l'organisation de travail du projet Junior. Nous avons utilisé Trello qui est une application conçue pour le management des projets. Très efficace, cet outil ergonomique permet également d'être aux commandes de plusieurs projets simultanément. Par ailleurs, il s'agit aussi d'un gestionnaire de tâches qui sert à mieux classer les projets à travers une liste de tâches.

Notre Trello : est un outil de gestion de projet en ligne, lancé en septembre 2011 et inspiré par la méthode Kanban de Toyota



3.5.2. VERSIONNING & WORKFLOW

Pour gérer les différentes modifications apportées dans le code et sauvegarder l'ensemble de notre travail, nous avons utilisé le logiciel de version de contrôle (VCS) Git, associé à son homologue en ligne GitHub . Il offre une interface graphique de gestion et de visualisation des actions opérées sur Git et permet une vue d'ensemble des différentes branches, commits, etc.

Pour organiser les différentes parties de l'application nous avons créé trois repositories :

Un repository Backend

- Contient la partie API développée avec Api-Platform

Un repository Backend

- Contient la partie API développée avec Api-Platform

Un repository Frontend

- Contient l'interface développée avec la librairie React Native

Sur les repositories Frontend et Backend, nous avons choisi d'articuler notre workflow autour de la

méthode de travail GitFlow. GitFlow permet de poser un cadre en standardisant la création de

branches et ainsi modéliser un workflow selon le type de tâche que l'on va faire.

Basés sur ce modèle, nos repositories possèdent chacun deux branches principales et des

branches secondaires :

Branches principales :

- Master : cette branche représente l'état du projet en production.
- Develop : chaque fonctionnalité terminée pendant un sprint sera fusionnée sur la branche develop et intégré plus tard dans la branche master.

Branches secondaires :

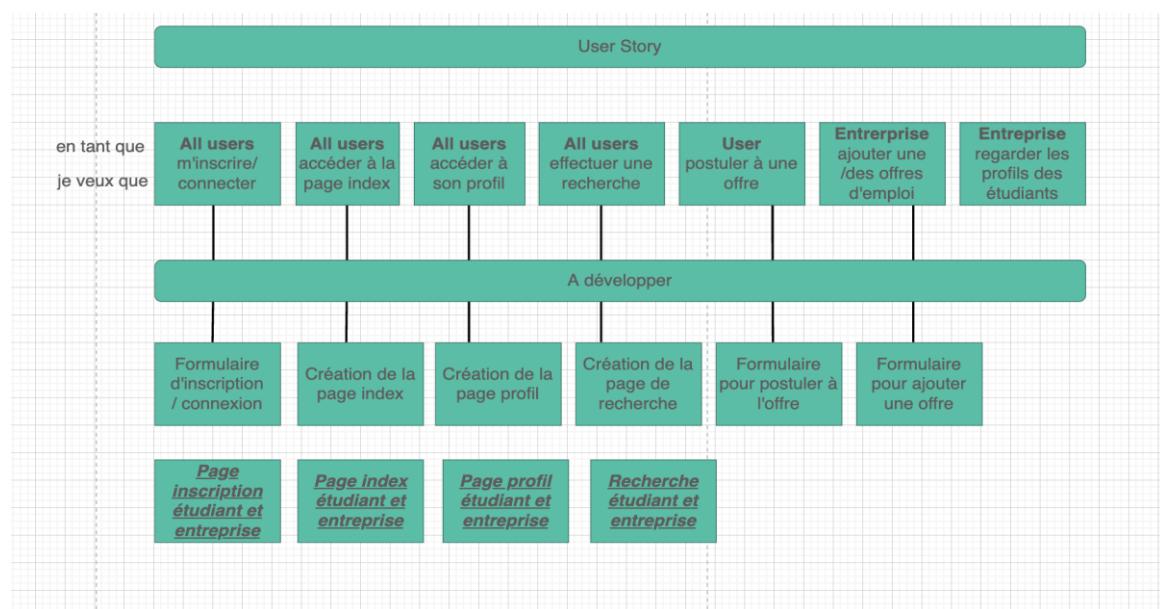
- feature : chaque nouvelle fonctionnalité sera développée sur une branche feature. La création d'une branche feature se fera toujours depuis la version la plus récente de la

branche develop et sera préfixée de cette manière - f/nom_de_la_branche. Une fois le sprint terminé, cette fonctionnalité sera Merge ou Rebase sur la branche develop.

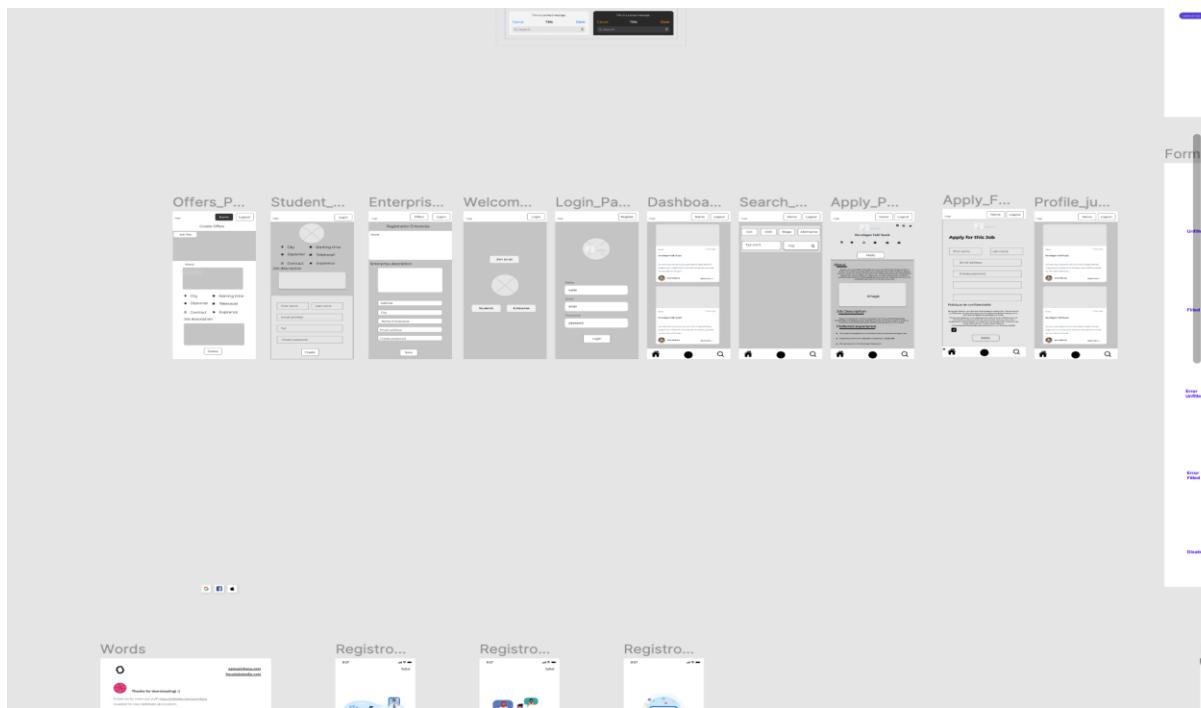
Pour chaque feature terminée nécessitant d'être fusionnée sur la

branche de référence Develop, nous avons fait le choix de faire valider ce processus par la création de Pull Request. Cela nous permet de faire du Code Review et de nous assurer, ou du moins minimiser, le risque de bug introduit suite à l'ajout de nouveau code.

3.1 USER STORY

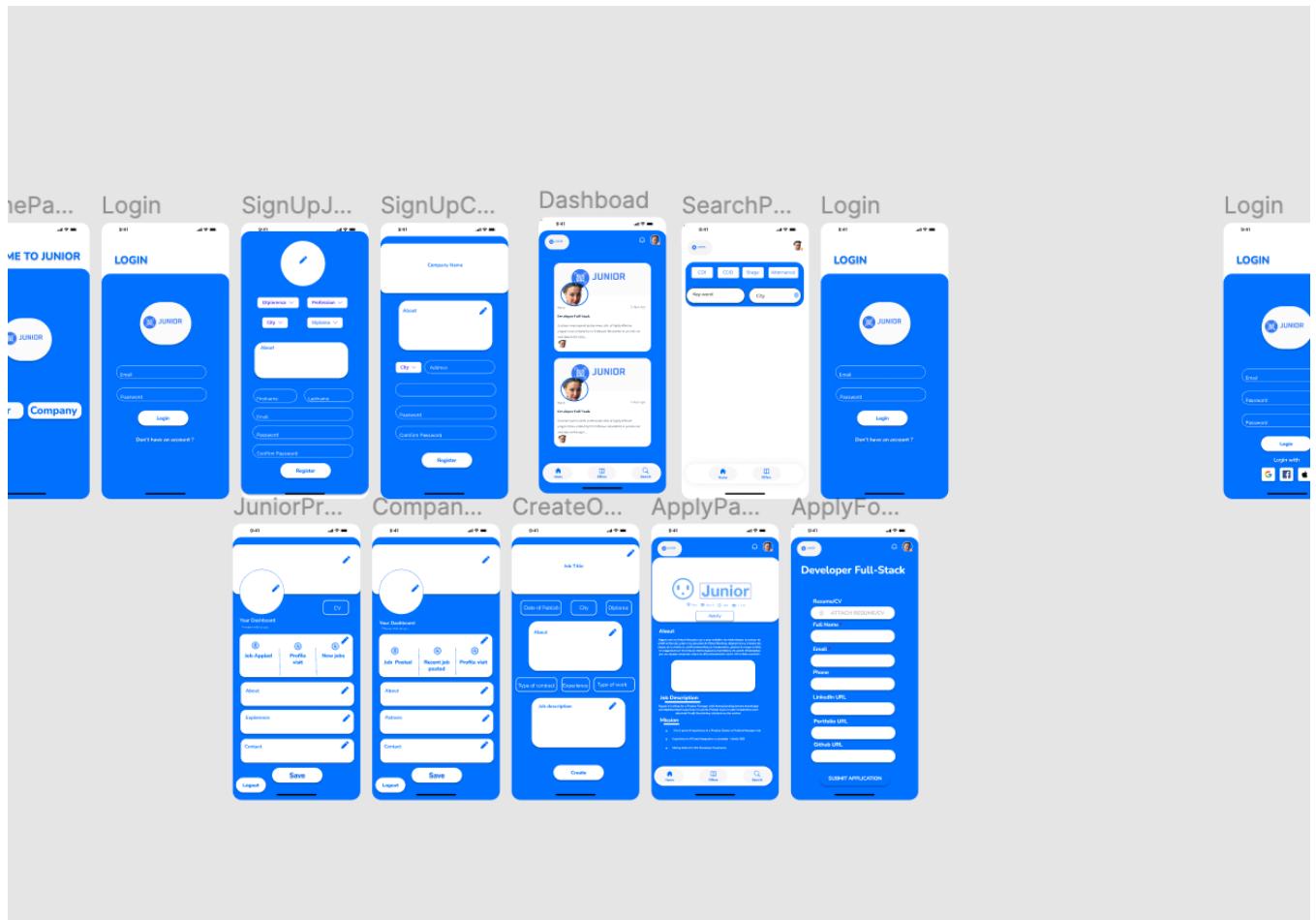


Wireframe the junior Application



3.2. UX

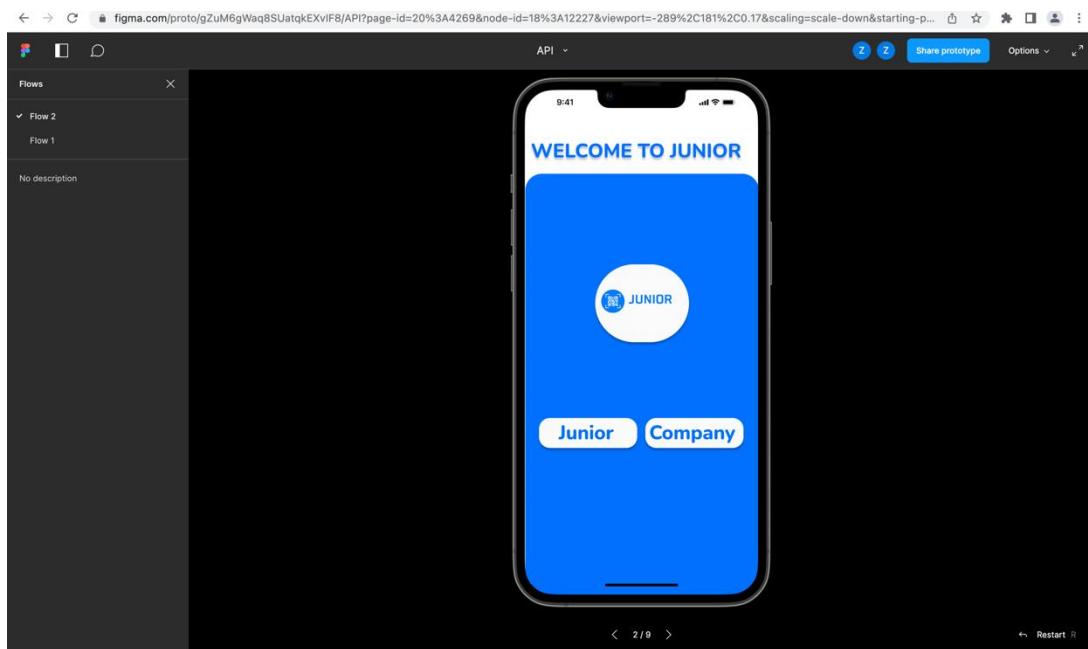
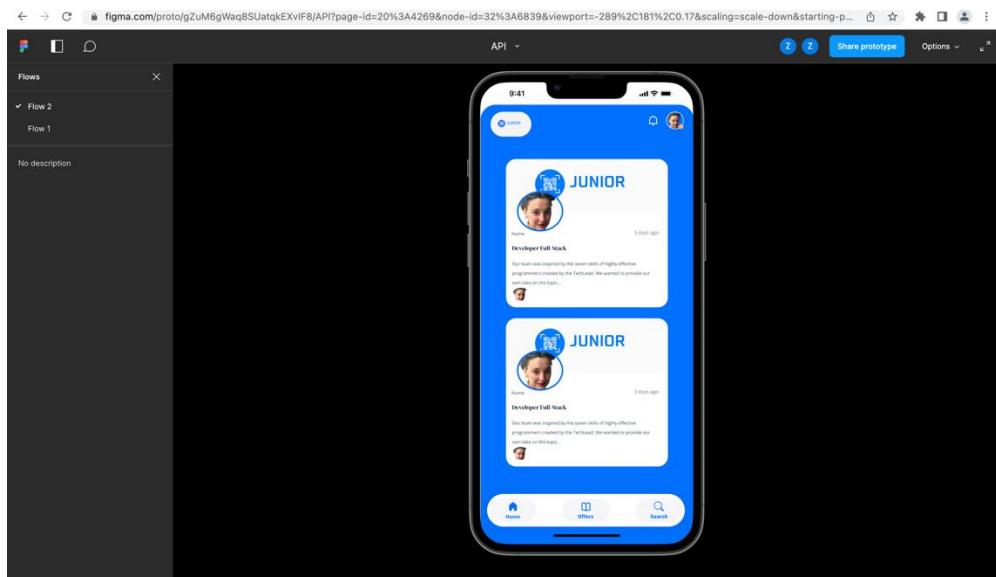
Design junior Application



MAQUETTE & PROTOTYPE

Ci-dessous la capture d'écran de ma maquette en prototype.

(https://www.figma.com/proto/gZuM6gWaq8SUatqkEXvIF8/API?page-id=20%3A4269&node_id=32%3A6839&viewport=-289%2C181%2C0.17&scaling=scale-down&starting-point-node-id=32%3A6839&showproto-sidebar=1)



3.4. MODELISATION DE LA BASE DE DONNEES

3.4.1. METHODE MERISE

MERISE (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise).

Née à la fin des années 1970 en France, elle a pour objectif de définir une démarche permettant la modélisation et la conception de S.I. Parmi les ressources informatiques de ces S.I., figurent en particulier les

fichiers de données, bases de données et système de gestion de bases de données (S.G.B.D.). C'est sur ce dernier point que la méthode MERISE nous a été utile, car c'est en se basant sur ses principes de modélisation que nous avons conçu la base de données de Junior.

3.4.2. MODELE CONCEPTUEL DE DONNEES (MCD)

Le MCD est une représentation graphique et structurée des informations mémorisées par un système d'information. Il est basé sur deux notions principales : les entités et les associations, d'où sa seconde appellation de schéma Entité/Association.

Pour schématiser le MCD de la BDD de Junior, nous avons dû passer par plusieurs étapes nécessaires à son élaboration :

Première étape : Définition des règles de gestion

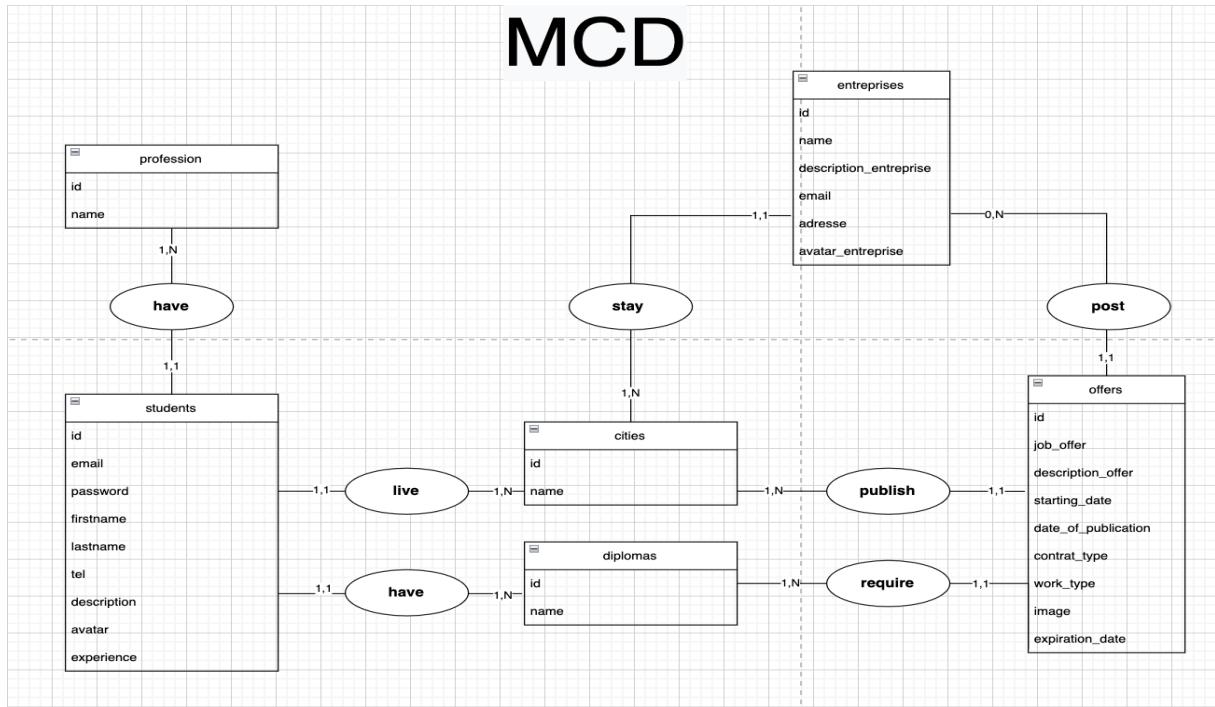
- Il nous a fallu recueillir les besoins potentiels des futurs utilisateurs de Junior. À partir de ces besoins, nous avons été en mesure d'établir les règles de gestion des données à conserver.

Deuxième étape : L'élaboration du dictionnaire des données

- Le dictionnaire des données nous a permis de regrouper toutes les données élémentaires que nous devons conserver dans la base et de définir certaines caractéristiques qui figureront dans le MCD. Parmi ces caractéristiques, l'on peut par exemple retrouver la référence d'une donnée et notamment un identifiant unique, sa désignation, son type, etc.

Troisième étape : L'élaboration du MCD

- Étape finale à sa conception, nous pouvons à partir des informations précédemment recueillies, créer chaque entité, unique et décrite par un ensemble de propriétés; Et leurs associations permettant de définir les liens et cardinalités entre les entités.



3.4.3. MODELE LOGIQUE DE DONNEES (MLD)

Le MLD est une étape de la modélisation permettant la transition entre le modèle conceptuel de données et le modèle physique de données (Cf. Modélisation BDD – Annexe 2). Le passage d'un MCD en MLD s'effectue selon quelques règles de conversion bien précises :

Première règle :

Une entité du MCD devient une relation, c'est-à-dire une table. Dans un SGBD de type relationnel, une table est une structure tabulaire dont

chaque ligne correspond aux données d'un objet enregistré et où chaque colonne correspond à une propriété de cet objet. Ces colonnes font notamment référence aux caractéristiques définies dans le dictionnaire de données du MCD.

Deuxième règle :

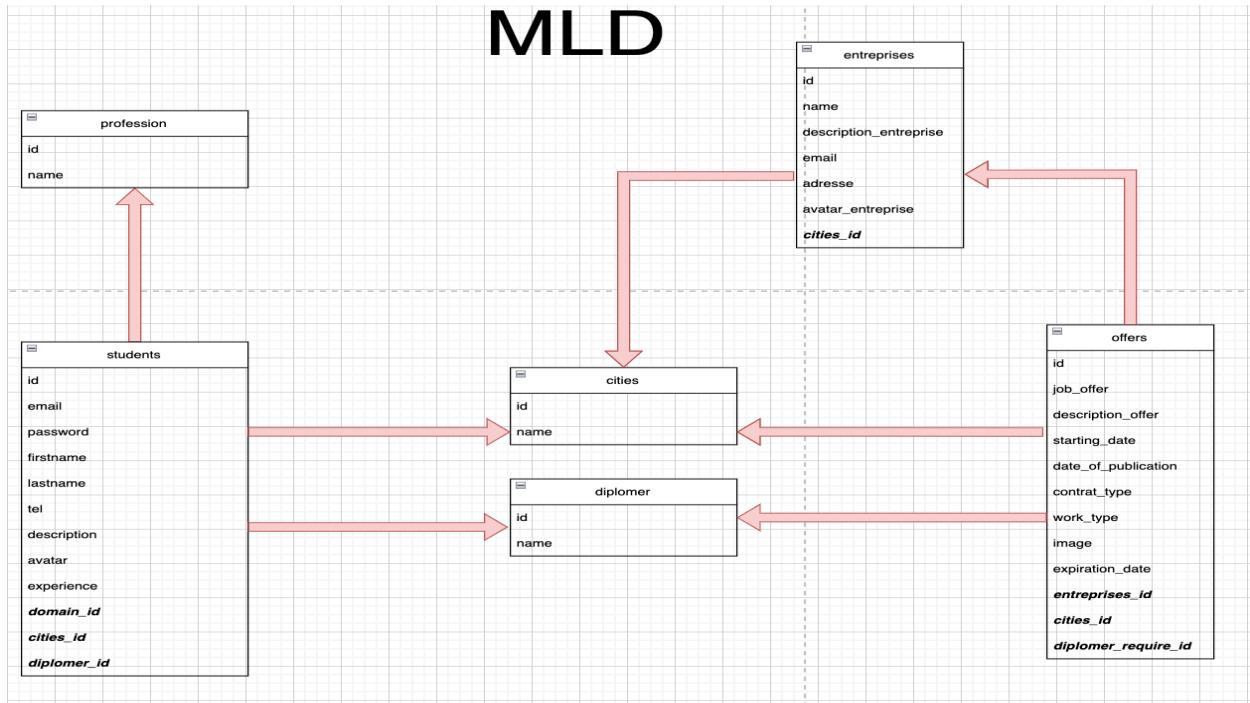
Les identifiants respectifs de chaque entité deviennent des clefs primaires et toutes les autres propriétés définies dans le MCD deviennent des attributs. Les clefs primaires permettent d'identifier de façon unique chaque enregistrement dans une table et ne peuvent pas avoir de valeur nulle.

Troisième règle :

Cardinalités (0,1) ou (1,1) vers (0,n) ou (1,n) : L'association disparaît et la clé de la relation relative à la cardinalité (0,n) ou (1,n) migre vers la relation relative à la cardinalité (0,1) ou (1,1). Cette clé est appelé « clé étrangère » .

Cardinalités (0,n) ou (1,n) vers (0,n) ou (1,n) : L'association devient une relation avec comme clé la concaténation des clés des 2 relations.

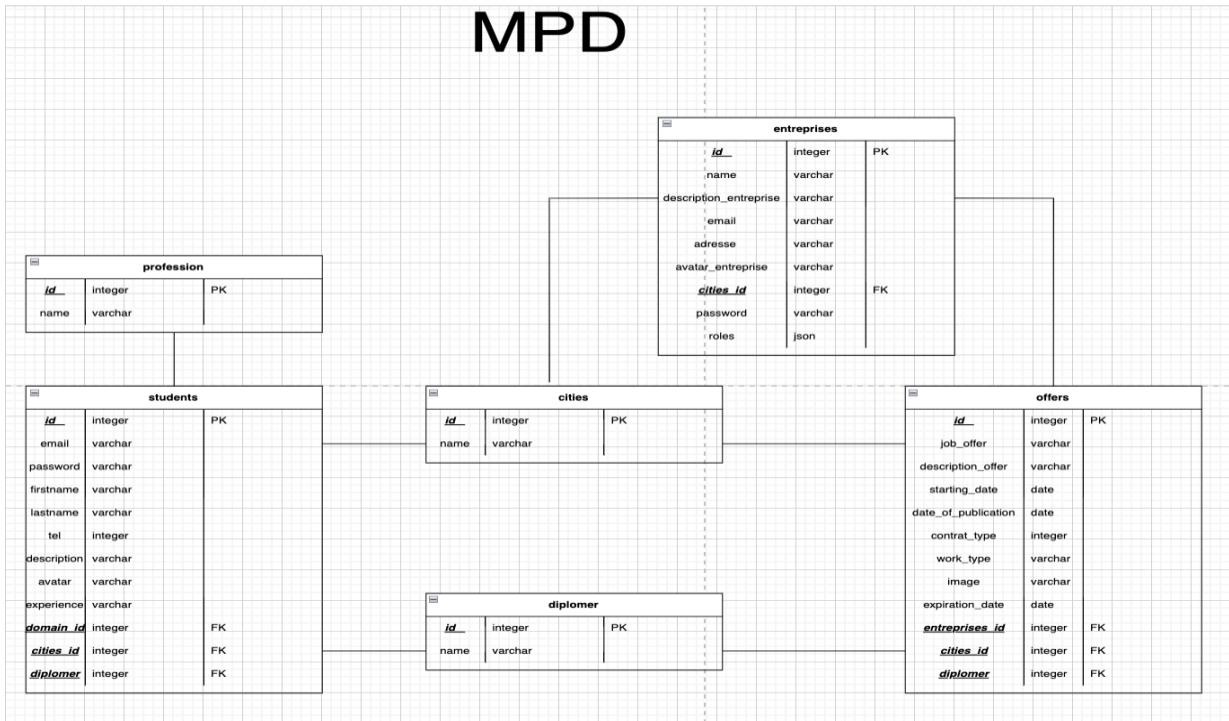
MLD



3.4.4. MODELE PHYSIQUE DE DONNEES (MPD)

Étant presque une formalité, le MPD consiste à l'implémentation des modèles générés précédemment dans le Système de Gestion de Base de Données (SGBD) utilisé. Dans notre cas, ce sera le SGBD MySQL couplé au moteur de stockage InnoDB qui permet la gestion des contraintes des clefs étrangères en base de données.

MPD



Dans le cadre de la conception de notre application mobile “Junior”, nous avons décidé d’utiliser Api-Platform.

Api-Platform est un framework écrit en PHP et basé sur Symfony qui permet de mettre en place simplement et rapidement une API REST, qui se base sur le design pattern MVC (Modèle, Vue, Contrôleur).

Pour la persistance des données L’ORM (Mapping objet-relationnel) Doctrine est livré avec la distribution Api-Platform. Doctrine est un moyen simple de persister et d’interroger des données grâce au pont fourni avec la distribution.

Doctrine Bridge est optimisé pour la performance et la commodité du développement. Lorsqu’on utilise Doctrine, Api-Platform optimise automatiquement les requêtes SQL générées en ajoutant les JOIN clauses appropriées. Il dispose notamment de nombreux filtres intégrés puissants.

Les SGBD (Système de Gestion de Base de Données) les plus populaires sont pris en charge (MySQL, MariaDB...).

Symfony Flex permet d’installer facilement Api-Platform depuis n’importe quelle application Symfony en utilisant le binaire Symfony.

Ligne de commande pour la création d’un projet Symfony :

```
symfony new nom-du-projet
```

ligne de commande pour l'installation d'Api-Platform :

```
composer require api
```

Après avoir installer ce dont nous avons besoin, je modifie le DATABASE_URL du fichier .env.

```
33 DATABASE_URL="mysql://root:root@127.0.0.1:8889/JuniorVersion01?serverVersion=5.7"
```

Cela permet à l'ORM Doctrine de savoir où se trouve ma base de données "JuniorVersion01" et d'y avoir accès.

La lignes suivante nous permet de créer la BDD dans notre SGBD :

```
symfony console doctrine:create:database
```

Puis nous pouvons créer nos table avec la ligne de commande :

```
symfony console make:entity
```

Après avoir crée des tables, j'ai effectué une migration pour mettre à jour la BDD avec la ligne de commande suivante :

```
symfony console make:migration
```

Cette commande génère un fichier de migration

```
$this->addSql('CREATE TABLE cities (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci');
$this->addSql('CREATE TABLE diplomas (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci');
$this->addSql('CREATE TABLE entreprises (id INT AUTO_INCREMENT NOT NULL, email VARCHAR(180) NOT NULL, roles JSON NOT NULL, password VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci');
$this->addSql('CREATE TABLE offers (id INT AUTO_INCREMENT NOT NULL, city_id INT NOT NULL, diploma_id INT NOT NULL, jobs VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci');
$this->addSql('CREATE TABLE profession (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci');
$this->addSql('CREATE TABLE refresh_tokens (id INT AUTO_INCREMENT NOT NULL, refresh_token VARCHAR(128) NOT NULL, username VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci');
$this->addSql('CREATE TABLE user (id INT AUTO_INCREMENT NOT NULL, city_id INT NOT NULL, profession_id INT NOT NULL, diploma_id INT NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci');
$this->addSql('ALTER TABLE offers ADD CONSTRAINT FK_DA4604278BAC62AF FOREIGN KEY (city_id) REFERENCES cities (id)');
$this->addSql('ALTER TABLE offers ADD CONSTRAINT FK_DA460427A99ACEBS FOREIGN KEY (diploma_id) REFERENCES diplomas (id)');
$this->addSql('ALTER TABLE user ADD CONSTRAINT FK_8D93D649BAC62AF FOREIGN KEY (city_id) REFERENCES cities (id)');
$this->addSql('ALTER TABLE user ADD CONSTRAINT FK_8D93D649FDEF8996 FOREIGN KEY (profession_id) REFERENCES profession (id)');
$this->addSql('ALTER TABLE user ADD CONSTRAINT FK_8D93D649A99ACEFB FOREIGN KEY (diploma_id) REFERENCES diplomas (id)').
```

On peut voir que L'ORM utilise le langage SQL, exemple : "CREATE TABLE cities..." = Créer une entité cities en BDD (JuniorVersion01), "ALTER TABLE offers ADD CONSTRAINT..." = Crée une clé étrangère contrainte (empêche les actions qui détruirait les liens entre tables).

Pour confirmer cette migration je rentre la ligne de commande suivante :

```
symfony console doctrine:migrations:migrate
```

Avec cette commande ma BDD est à jour.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
cities	Parcourir Structure Rechercher Insérer Vider Supprimer	33	InnoDB utf8mb4_unicode_ci	16,0 kio	-	
diplomas	Parcourir Structure Rechercher Insérer Vider Supprimer	18	InnoDB utf8mb4_unicode_ci	16,0 kio	-	
doctrine_migration_versions	Parcourir Structure Rechercher Insérer Vider Supprimer	9	InnoDB utf8_unicode_ci	16,0 kio	-	
entreprises	Parcourir Structure Rechercher Insérer Vider Supprimer	22	InnoDB utf8mb4_unicode_ci	48,0 kio	-	
offers	Parcourir Structure Rechercher Insérer Vider Supprimer	5	InnoDB utf8mb4_unicode_ci	64,0 kio	-	
profession	Parcourir Structure Rechercher Insérer Vider Supprimer	15	InnoDB utf8mb4_unicode_ci	16,0 kio	-	
refresh_tokens	Parcourir Structure Rechercher Insérer Vider Supprimer	5	InnoDB utf8mb4_unicode_ci	16,0 kio	-	
user	Parcourir Structure Rechercher Insérer Vider Supprimer	32	InnoDB utf8mb4_unicode_ci	80,0 kio	-	
8 tables	Somme	139	InnoDB utf8_general_ci	272,0 kio	0 o	

Ci-dessous la capture d'écran de la création de l'entity de l'api junior(entity)

```

API_junior > src > Entity > User.php
119     #[ORM\Column(type: 'integer')]
120     private $id;
121
122 /**
123 * @Vich\UploadableField(mapping="user_picture", fileNameProperty="photoFile")
124 * @var File
125 */
126 #[Assert\File(mimeTypes: ["image/png", "image/jpeg"], maxSize: '50M')]
127 private $photoFile;
128
129 #[ORM\Column(type: 'datetime', nullable: true)]
130     private $updatedAt;
131
132 #[Groups(["item"])]
133     private $jwtToken;
134
135 #[ORM\Column(type: 'string', length: 180, unique: true)]
136 #[Groups(["item"])]
137 #[Assert\Email(
138     message: 'L\'email {{ value }} n\'est pas valide.', // A enlever si site en anglais
139     )]
140     private $email;
141
142 #[ORM\Column(type: 'json')]
143

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER COMMENTS zsh - API_junior

```

Find the documentation at https://symfony.com/doc/current/testing.html#unit-tests
→ API_junior git:(AhceneAPIS) ✘ symfony php bin/phpunit
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

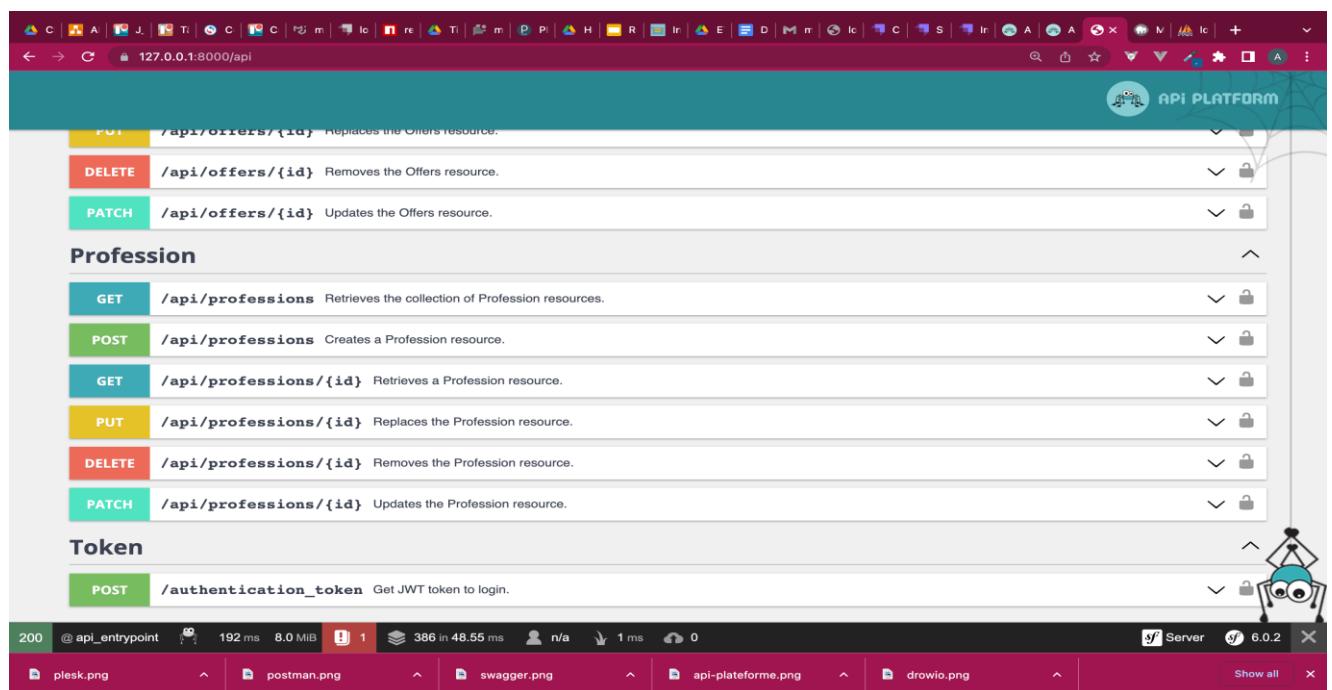
Testing .
Time: 00:00.011, Memory: 10.00 MB
OK (1 test, 1 assertion)
→ API_junior git:(AhceneAPIS) ✘

```

Ln 1, Col 1 Spaces: 4 UTF-8 LF PHP ⌂ Go Live ⌂ Prettier ⌂

user).

Ci-dessous la capture d'écran de notre création de route et la documentation de l'api avec swagger



The screenshot shows the Postman application interface. At the top, there's a navigation bar with tabs for Home, API, Collections, and Help. Below the navigation is a search bar and a toolbar with icons for file operations. The main content area displays a list of API endpoints categorized under 'User'. Each endpoint is shown with its method (e.g., GET, POST), path, and a brief description. The descriptions mention actions like 'Creates a Entreprises resource.', 'Removes the Entreprises resource.', etc. To the right of each endpoint, there are dropdown menus and lock icons. A sidebar on the right contains a cartoon character icon.

Ci-dessous la capture d'écran de notre mise en oeuvre de JWT

```

APIJunior > .env
# DATABASE_URL="mysql://toreapat:041219Tp&@127.0.0.1:3306/torea-patisserie_junior?serverVersion=5.5"
24
25  ##> doctrine/doctrine-bundle ####
26  # Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/
27  # configuration.html#connecting-using-a-url
28  #
29  # DATABASE_URL="sqlite:///kernel.project_dir/var/data.db"
30
31
32  # DATABASE_URL="mysql://toreapat:041219Tp&@127.0.0.1:3306/torea-patisserie_junior?serverVersion=5.5"
33  DATABASE_URL="mysql://root:root@127.0.0.1:8889/JuniorVersion01?serverVersion=5.7"
34
35
36  #DATABASE_URL="postgresql://symfony:ChangeMe@127.0.0.1:5432/app?serverVersion=13&charset=utf8"
37  ##< doctrine/doctrine-bundle ####
38
39  ##> nelmio/cors-bundle ####
40  CORS_ALLOW_ORIGIN='^https://(localhost|127\\0\\.0\\1)(:[0-9]+)?$'
41  ##< nelmio/cors-bundle ####
42
43  ##> lexik/jwt-authentication-bundle ####
44  JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
45  JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
46  JWT_PASSPHRASE=admin
47  JWT_TTL=3600
48  ##< lexik/jwt-authentication-bundle ####

```

The screenshot shows the VS Code code editor with the .env file open. The file contains environment variables for the APIJunior project. It includes configurations for Doctrine, a SQLite database, CORS, and the LexikJWTAuthenticationBundle. The code editor has a dark theme, and the .env file is highlighted in blue. The bottom status bar shows file statistics and various developer tools.

authentification pour l'api.

Ci-dessous la capture d'écran de entity user avec l'exécution de l'authentification avec JWT authentification (api-platefome) pour hâcher le mode passe.

	id	city_id	profession_id	diploma_id	updated_at	email	roles	password
1	7	NULL	NULL	NULL	NULL	Let@gmail.com	["RULE_USER"]	\$2y\$13\$0yJgF/hjatPyggILbVaBuc3dDvXVzD.EVdwVS1BY
2	12	2	2	2	2022-06-15 08:22:20	rahman@gmail.com	["RULE_USER"]	\$2y\$13\$W1dRwlFS/a.howrU91YC.oKxYUGkRouK.cOwPV
3	63	7	7	7	NULL	kiki@gmail.fr	["ROLE_USER"]	\$2y\$13\$7iLIP15i1jRcTywTMD.nsQzrcM/ZvhTwnoSPmEk
4	68	12	12	12	2022-06-15 12:42:11	mamam@gmail.com	["ROLE_USER"]	\$2y\$13\$/F6l8sSRrKlBMKNnjXdfFR.jmmlMmRSfhSpX4neB4R
5	73	20	15	20	2022-06-15 12:50:48	kakak@gmail.fr	["ROLE_USER"]	\$2y\$13\$NstuEC3/gH00yACCoPbj.upld0Myh3.tnVuHyox7G
6	76	22	22	22	2022-06-15 13:11:50	ugh@gmail.fr	["ROLE_USER"]	\$2y\$13\$Tjl3rUKLD5Gj3JBVrl/bQ.7TxhMDTs9mcljff3GW:
7	101	23	23	23	2022-06-16 07:20:39	work@gmail.com	["RULE_USER"]	\$2y\$13\$LghU.Ll9cnc4xCAcgFuug.Xqu.wDssepSXe32B49C
8	102	NULL	NULL	NULL	NULL	see@gmail.com	["ROLE_USER"]	\$2y\$13\$qdGrKzwdOajhu5.Jm0NGwEeEpi5IBO4CUkUlxpOf
9	103	NULL	NULL	NULL	NULL	mesee@gmail.com	["ROLE_USER"]	\$2y\$13\$Swq2NB1CWaI4KAwFJuCCv3J12qoYngQxnYoYKBkJ
10	104	NULL	NULL	NULL	NULL	myfr@gmail.com	["ROLE_USER"]	\$2y\$13\$cX6wwwWyBSr9mtT9AL8Quq7udy2kRRXkS30thw
11	106	NULL	NULL	NULL	NULL	alexif@gmail.com	["ROLE_USER"]	\$2y\$13\$0XCYJRIIMLBb32IBDjjo.BCGJeeZtBQ.IYAH1chhx.
12	107	NULL	NULL	NULL	NULL	emai@gmail.com	["ROLE_USER"]	\$2y\$13\$vMN7hd4gxPKYYhjEMmzUOYryRpncZpIclciH8C
13	108	NULL	NULL	NULL	NULL	szcpkczkp@gmail.fr	["ROLE_USER"]	\$2y\$13\$adR0FhZG.OB61Tojn06aTek0t5y3GB4bV6lRFcxH:
14	109	NULL	NULL	NULL	NULL	theo@herandez.com	["ROLE_USER"]	\$2y\$13\$yliuxphnHtW8Su4FwGoDs.UqGLNQOrFZuICzujkI9
15	110	NULL	NULL	NULL	NULL	ihefefef@gmail.fr	["ROLE_USER"]	\$2y\$13\$0IIIFxVbJZMYQMr5mA.vDy.hGhbq3stllcaqa-
16	111	NULL	NULL	NULL	NULL	inrh@gmail.fr	["ROLE_USER"]	\$2y\$13\$vgKomh4nzdu54AjB8r4q09SyWwUdmA90Luca15s
17	112	NULL	NULL	NULL	NULL	lookme@gmail.fr	["ROLE_USER"]	\$2y\$13\$Oos1iKzbkVu8CgXNW1X.zbuJ3QnZQlp7x8gmC
18	113	NULL	NULL	NULL	NULL	ahcen@example.com	["ROLE_USER"]	\$2y\$13\$Sxj/N8v0e1gVQN69K.lmRe01mAvnybbfZOYinqs
19	114	NULL	NULL	NULL	NULL	see@gmail.fr	["ROLE_USER"]	\$2y\$13\$Kmpt0ObBeInbCqMIWGViIeF5dMwtb9SKJhJRf9W
20	115	NULL	NULL	NULL	NULL	shiva@gmail.com	["ROLE_USER"]	\$2y\$13\$zqXu4PmIHeumeZlRvJU1.LDLejyk1rdYu7VR2l
21	116	NULL	NULL	NULL	NULL	djdz@gmail.com	["ROLE_USER"]	\$2y\$13\$IKieWGT88/yztA.YhsG8vveGvx5wWrJEUgAieWwr
22	117	25	25	25	2022-06-16 09:52:46	bjdzdbjz@gmail.fr	["ROLE_USER"]	\$2y\$13\$Adf4s4PaGkL2ISb8yQWR0kR3JLEsHaxqMLth3

Ci-dessous la capture d'écran des résultats des tests test de l'api (junior-api) sur postman

The screenshot shows the Postman interface with a successful API call response. The URL is `https://127.0.0.1:8000/authentication_token`. The response body is a JSON object containing a token and refresh token.

```

{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpX01oJEZNTUSMzUzNTEsIm4cIi6MTYNTkzODh1MSwicm9sZXMiOsiUk9MRV9VU0VS10sInVzXJuYW1ljojd29ya0BnbWFpbC5mcij9.eyJnR8RjJu_Iu_RaujLNcWj0RpqfViencnNtQjhzb36XIKD0-hb011EBYzVctWsA8NF9Qj0gjv0-axQuBF02Mbd1WaTVLBA-8gdi1u21tvKzQts9kPwAk_555JL3F3NFu0NzQ1giU0gHPNzSbf9Qgw8816uE-EZ_313cNALMs09S-Hc36Z6kt0pnbkch-Q1-H7-1MTc_55x2YkwxAiX0z5N1loab0igJ_Lc4ohwm6G17KN-D10HFVFJlmh24yC84bvw9qkROZC1ch1LV9TwzQjgH91ASh09wfKPfipjz7MpxRDNmVYyPxuZRJ_naq9ju8WhRsomQDMNu5mSpfNxrt8s9aixw31T26s8PFYh0Axq1-kc0f9FuonFrKhFgAtUP3NYflgUL-aNMEx77svVMyk0d_AfK3k14j2Qmz-5i0T8lP0gxv3itb0LTw-zRbm168gEAs_zFMk1QrqghvLAP_Snehp1yEcQhWF867NHlHwn5ULTtGsvjfhL_PKRpWhcmzIsyngocJ3tA8DvTCg14Bj0gGMRpTzK4oibrLs6NZUeYBmrccyZB5zNDH283Zn9jvcFBKaiuyPjQ_IK2McsV2wPz6Cutz01YdponmfgIxgM6_0ltqzIAWmAHS8hoftz-VasS0",
  "refreshToken": "7bb28b756e26dbe262277a0cb002d9c1016aa9dce371ffcc3d567964e2659b98e9292008470289741d41bc3650f4577ec5eb7e86deea29e535851f762966c8eb0"
}

```

Ci-dessous la capture d'écran du Test Unitaire sur symfony avec api-platform.

The screenshot shows the VS Code interface with the PHPUnit test results in the terminal. The command run was `symfony console make:test TestCase SpamCheckerTest`.

```

Context.js .phpunit.result.cache
APL_junior > .phpunit.result.cache
1 {"version":1,"defects":[],"times":{"App\Tests\SpamCheckerTest::testSomething":0.003}}


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER COMMENTS zsh - API_junior + × □ ^ ×

Package operations: 0 installs, 0 updates, 1 removal
  - Removing symfony/test-pack (1.0.10)
Generating optimized autoload files
composer/package-versions-deprecated: Generating version class...
composer/package-versions-deprecated: ...done generating version class
114 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

Info from https://repo.packagist.org: #StandWithUkraine
Executing script cache:clear [OK]
Executing script assets:install public [OK]

→ API_junior git:(AhceneAPIS) x symfony console make:test TestCase SpamCheckerTest
created: tests/SpamCheckerTest.php

Success!

Next: Open your new test class and start customizing it.
Find the documentation at https://symfony.com/doc/current/testing.html#unit-tests
→ API_junior git:(AhceneAPIS) x symfony php bin/phpunit
PHPUnit 9.5.13 by Sebastian Bergmann and contributors.

Testing
.
Time: 00:00.011, Memory: 10.00 MB
OK (1 test, 1 assertion)
→ API_junior git:(AhceneAPIS) x

```

Review

Livrable (<http://127.0.0.1.8000/>)

Plesk est une interface de gestion de serveur payante, mais j'ai pu y avoir un accès gratuit en tant qu'étudiant à la Plateforme. J'ai donc décidé d'y déployer mon API dans le cadre de mon projet d'application mobile.

Pour cela j'ai procédé comme suit

- 1) J'ai ajouté mon dépôt git sur plesk
- 2) J'ai choisi le mode de déploiement automatique, cela signifie que toutes les étapes se font automatiquement.
- 3) Pour la connexion au dépôt git distant j'ai créé une clé publique SSH

Code location

 **Remote repository**
Your code is hosted online (a cloud service like GitHub, GitLab, or Bitbucket, or your own server).
Plesk will **pull** code from there.

 **Local repository**
Your code is on the computer to which Plesk wouldn't be able to connect. For example it's your laptop or some server unavailable outside of your LAN.
You will **push** code to the server with Plesk yourself.

Repository URL *

git@github.com:torea-patisser/API_junior.git

Both HTTP(S) and SSH protocols are supported

SSH public key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQACQ3uOq8WyMPDckExgicCwmbEwy
Can5s6gcM4US9Tl+bWi11nmQ8NreDKUzRkKh3HRMkL7ZIS9+ccGFBK3wXn
Dhm2XCEw9G8NyHVi+jKUpSseCpKcqFDns3NNoZT2Etjo4jfgNuPODEINpp
```

This is the public part of the SSH key used for authorization in the remote repository. It must be added to the remote service.

Repository name *

API_junior.git

Specify a name that is unique within a domain.

Deployment settings

Deployment mode *

Automatic Manual Disabled

Files will be deployed to the production site as soon as they are available in the Plesk repository.

Server path *

/htdocs



Directory on the server where files will be deployed.

Enable additional deployment actions

Specify shell commands to be run every time upon deployment.

api.torea-patissier.students-laplateforme.io • Active ▾

Website at api.torea-patissier.students-laplateforme.io/public/ IP address: 93.90.200.103 System user: torea-patissier

Hosting Settings Open in web Preview Description Move domain

File Manager Mail Databases [torea-patissier_API](#) [Open](#)

API_junior.git Branch: toreaV2; Automatic deploy to /api.torea-patissier.students-laplateforme.io [Pull Updates](#)

[Show Less](#)

Web Hosting Access	FTP Access	Hosting Settings
PHP Composer	SSL/TLS Certificates Security can be improved	Git Enabled
PHP Settings Version 8.0.20	Applications	Node.js
File Manager	Web Statistics SSL/TLS	DNS Settings
Mail Settings	Password-Protected Directories	Website Copying
Logs	Web Users	Remove Subdomain

Ci-dessous la capture d'écran de notre déploiement de l'api junior sur plesk

Le nom de la Database

Déploiement automatique

Le nom de la branche sur laquelle l'API est déployé (toreaV2)

The screenshot shows the configuration for a Git repository named 'API_junior.git'. The URL is set to 'git@github.com:torea-patissier/API_junior.git'. The active branch is 'toreaV2'. Below this, a section titled 'Latest commits' lists two recent commits: one from 2022-06-21 17:14 labeled 'BUG' and another from 2022-06-21 16:31 labeled 'me'. There is a link to 'show more' commits. A large grey button labeled 'Pull now' is visible. Under the heading 'Deployment', it says 'toreaV2 branch automatically to /api.torea-patissier.students-laplateforme.io'. A button labeled 'Deploy now' is present. At the bottom right are several small icons: a trash can, a gear, and a circular arrow.

NPM : GESTION DES DÉPENDANCES CÔTÉ FRONT-END

Parmi les différentes fonctionnalités développées, certaines, nécessaires au projet, dépendent de l'utilisation de librairies externes. Également nommées dépendances de projet, ces librairies permettent l'implémentation de fonctionnalités déjà créées par d'autres développeurs, nous faisant ainsi

gagner en productivité et nous évitant de réinventer la roue. Voici quelques exemples de librairies utilisées dans le projet :

- Axios qui est un Client HTTP simplifiant les requêtes en BDD
- React Navigation qui permet le routing et le partage de données entre plusieurs écrans

Pour gérer ces dépendances, nous utilisons l'outil NPM. Ce gestionnaire de paquets Node qui est fourni avec le logiciel et qui facilite tout développement Node. Depuis des années, Node a été largement utilisé par les développeurs [JavaScript](#) pour partager des outils, installer divers modules et gérer leurs dépendances. Sachant cela, il est extrêmement important pour les personnes travaillant avec Node.js de comprendre ce qu'est npm. Cet utilitaire permet d'installer et de désinstaller des paquets, de gérer les versions et les dépendances nécessaires à l'exécution d'un projet (Cf. Gestion des dépendances – Annexe 6).

3.5.4. COMPOSER : GESTION DES DEPENDANCES CÔTÉ BACK-END

4. ENVIRONNEMENT TECHNIQUE

4.1. SPECIFICITES TECHNIQUES : BACKEND

4.1.1. API RESTFUL

De par la nature du projet, une application mobile multiplateforme, nous avons fait le choix concernant le traitement et la gestion des données, de nous tourner vers la création d'une API.

Une API est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d' applications.

Elle est parfois considérée comme un contrat entre un fournisseur d'informations et un utilisateur d'informations, qui permet de définir le contenu demandé au consommateur (l'appel) et le contenu demandé au producteur (la réponse).

Plusieurs architectures d'API existent. Parmi elles, on retrouve l'architecture REST. REST est un ensemble de contraintes architecturales. Il ne s'agit ni d'un protocole, ni d'une norme. Les développeurs d'API peuvent mettre en œuvre REST de nombreuses manières. Lorsqu'un client émet une requête par le biais d'une API RESTful, celle-ci transfère une représentation de l'état de la ressource au demandeur ou point de terminaison. Cette information, ou représentation, est fournie via le protocole HTTP dans l'un des formats suivants : JSON

(JavaScript Object Notation), HTML, XLT, Python, PHP ou texte brut. Le langage de programmation le plus communément utilisé est JSON, car, contrairement à ce que son nom indique, il ne dépend pas d'un langage et peut être lu aussi bien par les humains que par les machines. Autre point à retenir : les en-têtes et paramètres jouent également un rôle majeur dans les méthodes HTTP d'une requête HTTP d'API RESTful, car ils contiennent des informations d'identification importantes concernant la requête (métadonnées, autorisation, URI, mise en cache, cookies, etc.). Il existe des en-têtes de requête et des en-têtes de réponse. Chacun dispose de ses propres informations de connexion HTTP et codes d'état.

Une API RESTful doit remplir les critères suivants :

Client-serveur separation

Cette séparation permet au client de s'occuper uniquement de la récupération et de l'affichage de l'information et permet au serveur de se concentrer sur le stockage et la manipulation des données. Du moment que le serveur et le client structurent leur communication selon les lignes directrices architecturales REST, en utilisant le protocole http, ils pourront communiquer entre eux.

Stateless

Des communications client-serveur stateless, c'est-à-dire que les informations du client ne sont jamais stockées entre les requêtes GET, qui doivent être traitées séparément, de manière totalement indépendante.

Cacheable

La possibilité de mettre en cache des données afin de rationaliser les interactions client-serveur.

Uniform Interface

Une interface uniforme entre les composants qui permet un transfert standardisé des informations Cela implique que

- les ressources demandées soient identifiables et séparées des représentations envoyées au client.
 - les ressources puissent être manipulées par le client au moyen de la représentation reçue, qui contient suffisamment d'informations.
-
- les messages autodescriptifs renvoyés au client contiennent assez de détails pour décrire la manière dont celui-ci doit traiter les informations.

- l'API possède un hypertexte/hypermédia, qui permet au client d'utiliser des hyperliens pour connaître toutes les autres actions disponibles après avoir accédé à une ressource.

Layered System

Un système à couches, invisible pour le client, qui permet de hiérarchiser les différents types de serveurs (pour la sécurité, l'équilibrage de charge, etc.) impliqués dans la récupération des informations demandées.

Code on Demand

Du code à la demande (facultatif), c'est-à-dire la possibilité d'envoyer du code exécutable depuis le serveur vers le client (lorsqu'il le demande) afin d'étendre les fonctionnalités d'un client.

Bien que l'API REST doive répondre à l'ensemble de ces critères, elle est considérée comme étant plus simple à utiliser qu'un protocole tel que SOAP (Simple Object Access Protocol), qui est soumis à des contraintes spécifiques, dont la messagerie XML, la sécurité intégrée et la conformité des transactions, ce qui le rend plus lourd et moins rapide.

Puisque REST est un ensemble de directives mises en œuvre à la demande, les API REST sont plus rapides et légères, et offrent une évolutivité accrue.

4.2. SPÉCIFICITÉS TECHNIQUES : FRONT-END

4.2.1. REACT NATIVE

Un framework de développement mobile est l'un des éléments les plus essentiels pour créer des applications mobiles riches en fonctionnalités et hautement performantes. Le choix du cadre framework a un impact direct sur la qualité du développement et l'expérience des utilisateurs. React Native est l'un des principaux frameworks de développement mobile utilisés par les développeurs du monde entier aujourd'hui. Il offre des fonctionnalités robustes et un nombre important de fonctions puissantes pour les applications iOS et Android. React Native est un projet soutenu par une large communauté de développeurs, et la plateforme encourage les développeurs à contribuer à l'amélioration du framework. De nombreux développeurs qui cherchent à reprendre et à utiliser React Native peuvent utiliser pleinement la technologie communautaire.

React Native est connu pour fournir un niveau de performance élevé grâce à l'utilisation de modules et de contrôle natif. React Native est connecté à des composants natifs pour les deux systèmes d'exploitation et est pratique pour générer du code API natif. Les performances des applications React Native sont optimales car elles utilisent différents threads pour les API natives et l'interface utilisateur.

Dans premier temps

Installation du framword react-native avec expo avec utilisation de la documentations a partir des (<https://docs.expo.dev/get-started/create-a-new-app/>).

Commandes suivantes :

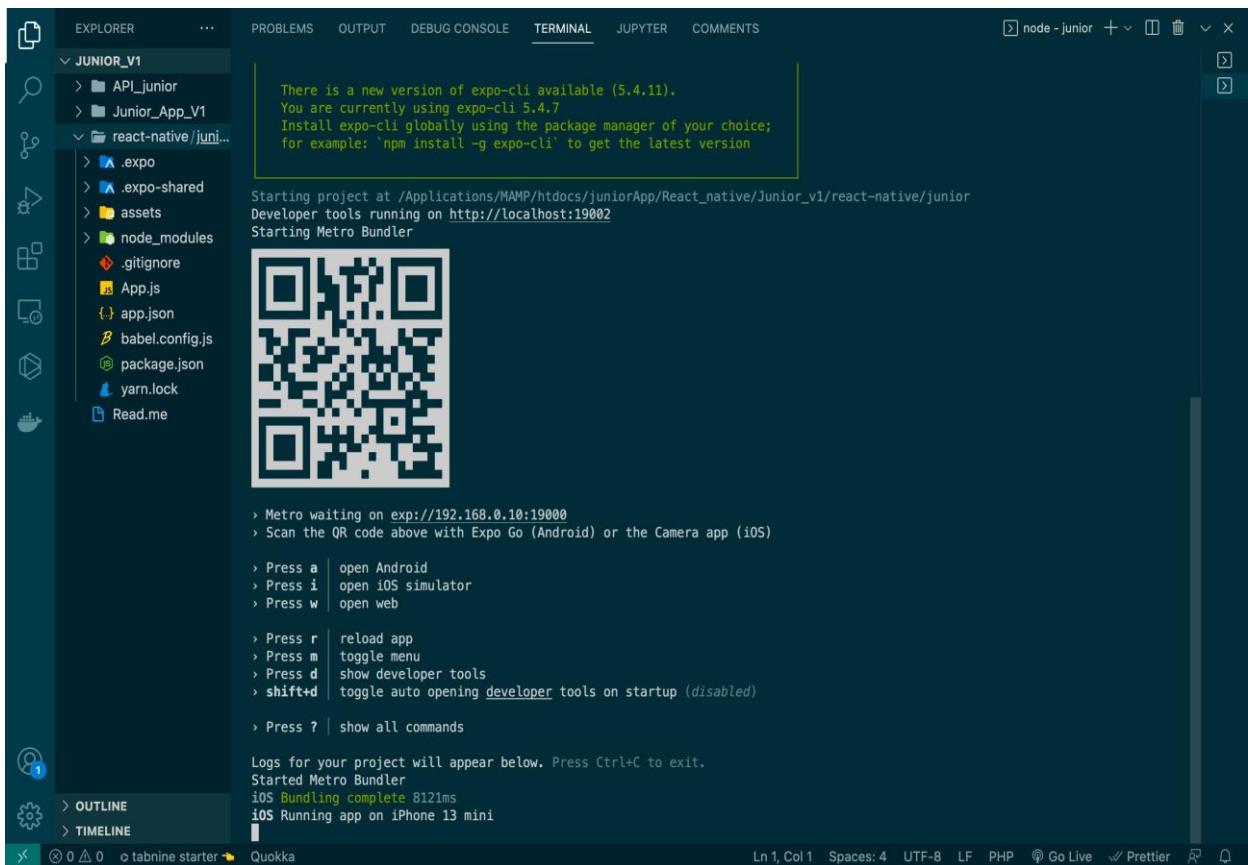
```
expo init JuniorApp
```

```
cd JuniorApp
```

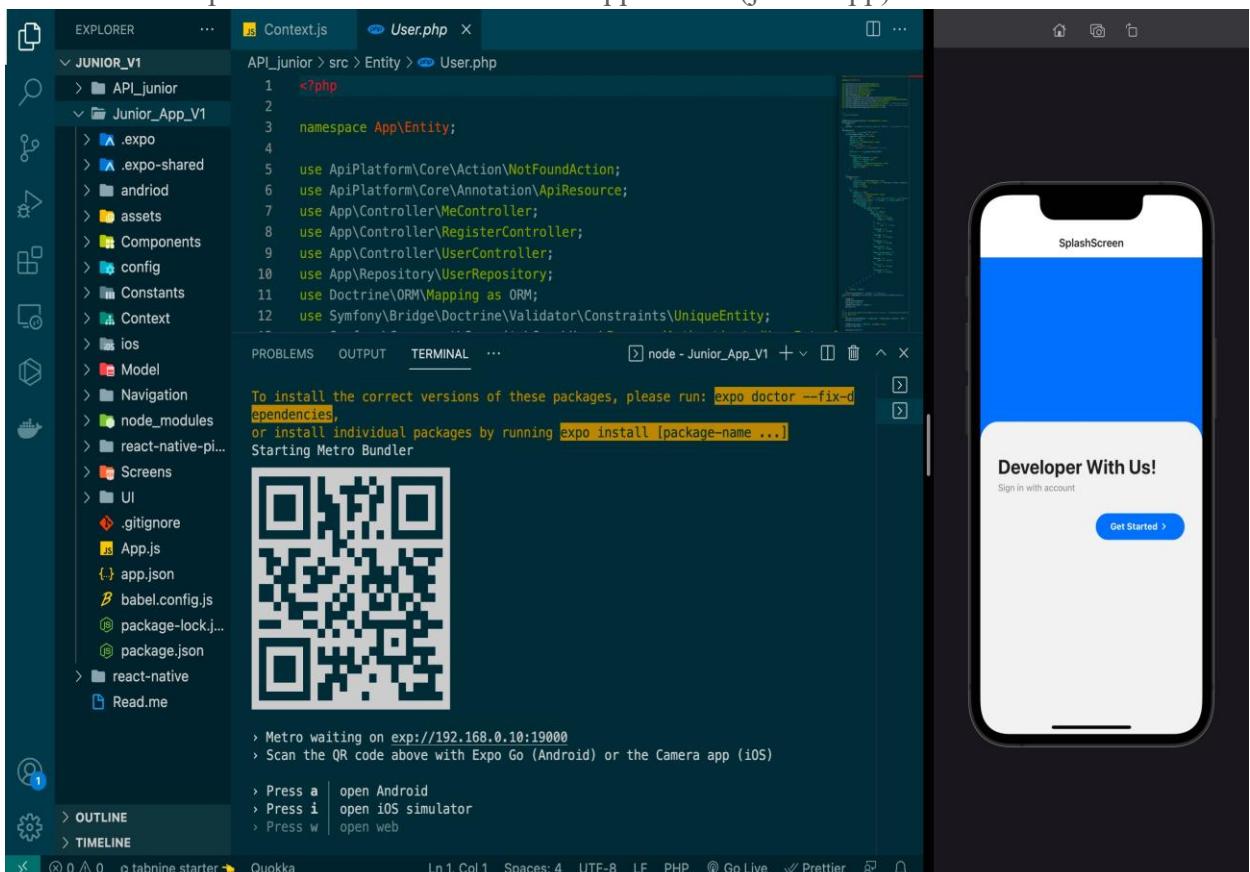
Expo start

- . Création de composant, screen avec react-native
- . Création de navigation en utilisant react-navigation
- . Faire un appel de l'api pour connecter le front et le back de notre application (juniorApp)
- . Utiliser Xcode pour tester, debugger l'application (juniorApp)
- . Création de l'authentification avec useContext
- . Stockage des informations avec Asyncstorage
- . Tester l'application (juniorApp)

Ci-dessous la capture d'écran du lancement de juniorApp avec expo cli.



Ci-dessous la capture d'écran de test de notre application (juniorApp) en

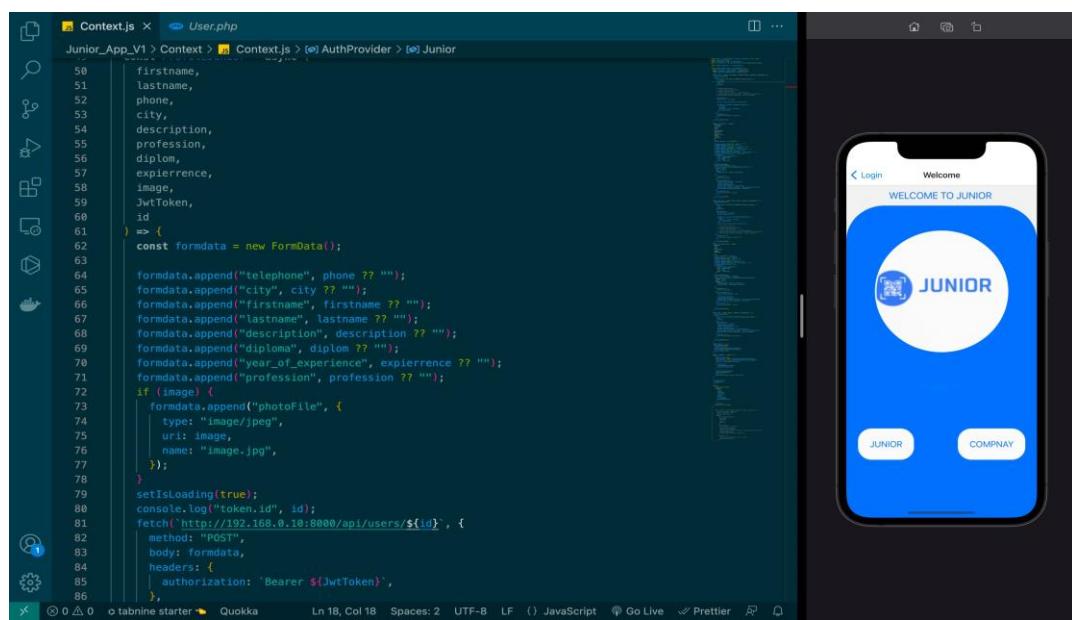


utilisant Xcode pour lancer l'application sur iPhone.

Ci-dessous la capture d'écran d'exécution de la react-navigation pour définir mes stack screen et utiliser les différentes naviagation (Stack, Tab et Drawer naviagation).

Nmp install @react-naviagation/native

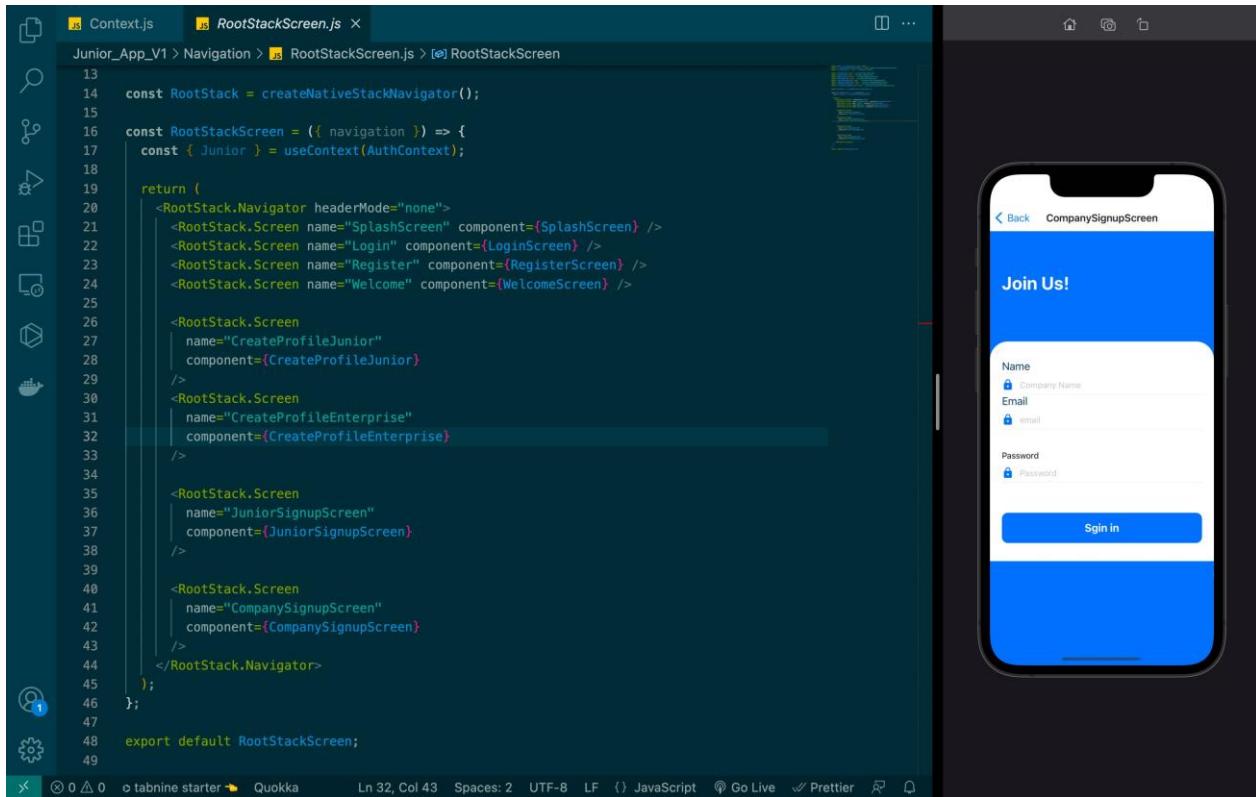
Expo install react-native-screens react-native-safe-area-context



The screenshot shows a developer's environment with two main windows. On the left is a code editor displaying a file named 'Context.js'. The code is a function that takes an object with properties like 'firstname', 'lastname', 'phone', 'city', 'description', 'profession', 'diplom', 'expierrence', 'image', 'JwtToken', and 'id'. It then creates a new FormData object, appends various fields to it, and performs a POST request to 'http://192.168.0.10:8000/api/users/\${id}'. On the right is a mobile application running on an iPhone X simulator. The app has a blue background with a white circular logo containing the word 'JUNIOR'. Below the logo, there are two buttons labeled 'JUNIOR' and 'COMPNEY'. The top status bar of the simulator shows 'Login' and 'Welcome'.

```
50   firstname,
51   lastname,
52   phone,
53   city,
54   description,
55   profession,
56   diplom,
57   expierrence,
58   image,
59   JwtToken,
60   id
61 } => {
62   const formdata = new FormData();
63
64   formdata.append("telephone", phone ?? "");
65   formdata.append("city", city ?? "");
66   formdata.append("firstname", firstname ?? "");
67   formdata.append("lastname", lastname ?? "");
68   formdata.append("description", description ?? "");
69   formdata.append("diploma", diplom ?? "");
70   formdata.append("year_of_experience", expierrence ?? "");
71   formdata.append("profession", profession ?? "");
72   if (image) {
73     formdata.append("photoFile", {
74       type: "image/jpeg",
75       uri: image,
76       name: "image.jpg",
77     });
78   }
79   setIsLoading(true);
80   console.log("token.id", id);
81   fetch(`http://192.168.0.10:8000/api/users/${id}`, {
82     method: "POST",
83     body: formdata,
84     headers: {
85       authorization: `Bearer ${JwtToken}`,
86     },

```



```
Context.js RootStackScreen.js
Junior_App_V1 > Navigation > RootStackScreen.js > RootStackScreen
13
14 const RootStack = createNativeStackNavigator();
15
16 const RootStackScreen = ({ navigation }) => {
17   const { Junior } = useContext(AuthContext);
18
19   return (
20     <RootStack.Navigator headerMode="none">
21       <RootStack.Screen name="SplashScreen" component={SplashScreen} />
22       <RootStack.Screen name="Login" component={LoginScreen} />
23       <RootStack.Screen name="Register" component={RegisterScreen} />
24       <RootStack.Screen name="Welcome" component={WelcomeScreen} />
25
26       <RootStack.Screen
27         name="CreateProfileJunior"
28         component={CreateProfileJunior}
29       />
30       <RootStack.Screen
31         name="CreateProfileEnterprise"
32         component={CreateProfileEnterprise}
33       />
34
35       <RootStack.Screen
36         name="JuniorSignupScreen"
37         component={JuniorSignupScreen}
38       />
39
40       <RootStack.Screen
41         name="CompanySignupScreen"
42         component={CompanySignupScreen}
43       />
44     </RootStack.Navigator>
45   );
46
47
48 export default RootStackScreen;
```

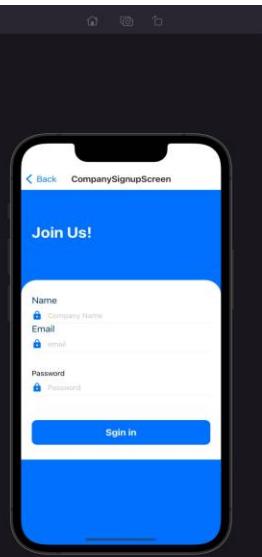
Ci-dessous la capture d'écran d'exécution de useConext qui inclut toutes mes fonctions qui permettent d'appeler de l'api et de passer les variables sur tous les composants et l'écran.

Ci-dessous la capture d'écran d'exécution de Asyncstorage pour stocker mes données et rafraîchir chaque itération avec l'aide de useEffect.

NOTE:

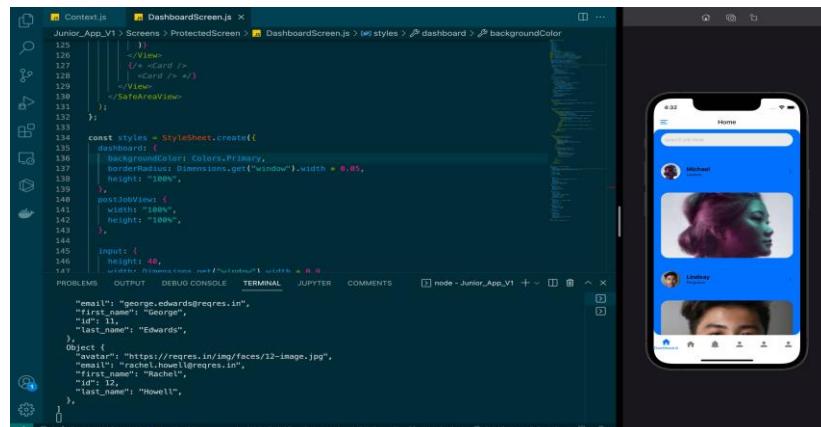
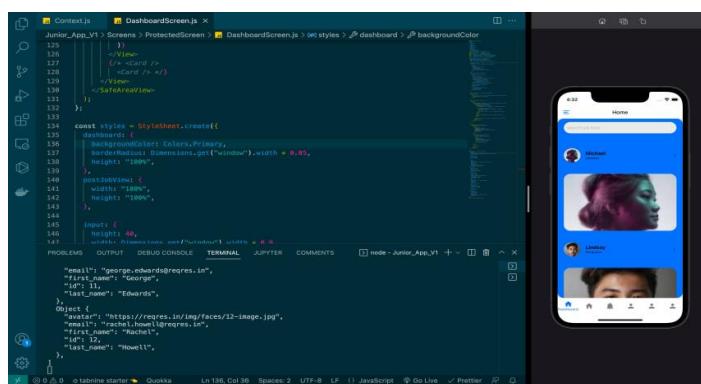
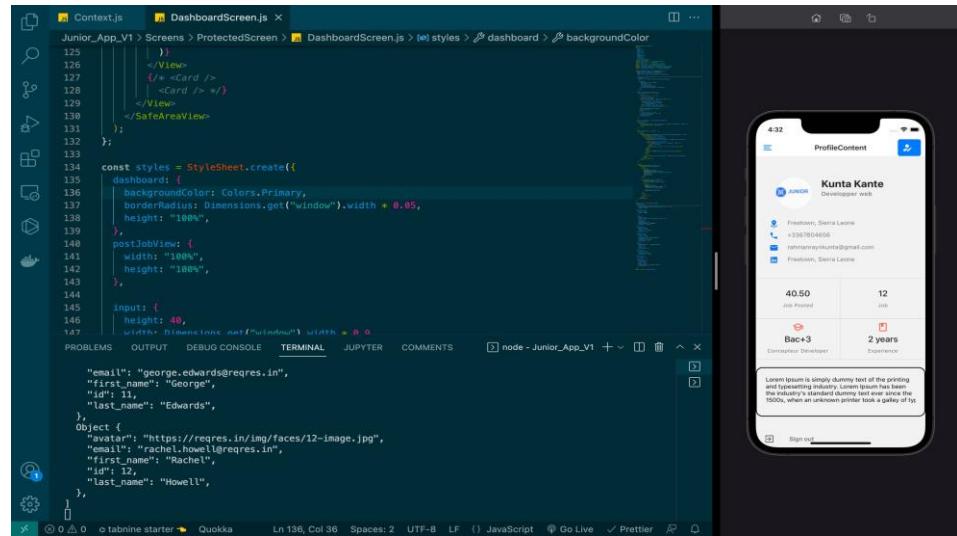
Le stockage asynchrone ne peut stocker que des données de chaîne, donc pour stocker des données d'objet, vous devez d'abord les sérialiser. Pour les données qui peuvent être sérialisées en JSON, vous pouvez utiliser JSON.stringify() lors de l'enregistrement des données et JSON.parse() lors du chargement des données

expo install @react-native-async-storage/async-storage



The screenshot shows a code editor on the left and a mobile application interface on the right. The code editor displays the file `Context.js` with several lines of JavaScript code. The mobile application interface shows a screen titled "Join Us!" with input fields for Name, Company Name, Email, and Password, and a "Sign In" button.

```
Junior_App_V1 > Context > Context.js > (e) AuthProvider > (e) Junior
186     setLoading(false);
187   };
188
189   const login = async (email, password, navigation) => {
190     setLoading(true);
191     axios
192       .post("http://192.168.0.10:8000/authentication_token",
193         {
194           email,
195           password,
196         }
197       )
198       .then((res) => {
199         let userInfo = res.data;
200         setUserInfo(userInfo);
201         setUserToken(userInfo.token);
202         console.log("User Token" + userInfo.token);
203         console.log("userInfo", JSON.stringify(userInfo));
204         AsyncStorage.setItem("userInfo", JSON.stringify(userInfo));
205         AsyncStorage.setItem("userToken", userInfo.token);
206       })
207       .catch((err) => {
208         console.log(`Login error ${err}`);
209       });
210     setLoading(false);
211   };
212   const logout = () => {
213     setLoading(true);
214     setUserToken(null);
215     AsyncStorage.removeItem("userInfo");
216     AsyncStorage.removeItem("userToken");
217     setLoading(false);
218   };
219
220   const isLoggedIn = async () => {
221     try {
222       setLoading(true);
223       let userInfo = await AsyncStorage.getItem("userInfo");
224     }
225   };
226
227   const [name, setName] = useState("");
228   const [companyName, setCompanyName] = useState("");
229   const [email, setEmail] = useState("");
230   const [password, setPassword] = useState("");
231
232   const handleNameChange = (text) => {
233     setName(text);
234   };
235
236   const handleCompanyNameChange = (text) => {
237     setCompanyName(text);
238   };
239
240   const handleEmailChange = (text) => {
241     setEmail(text);
242   };
243
244   const handlePasswordChange = (text) => {
245     setPassword(text);
246   };
247
248   const handleSignIn = () => {
249     if (!name || !email || !password) {
250       alert("Please fill in all fields");
251       return;
252     }
253     login(email, password, navigation);
254   };
255
256   return (
257     <View>
258       <Text>Join Us!</Text>
259       <Form>
260         <Text>Name</Text>
261         <Input type="text" value={name} onChange={handleNameChange}></Input>
262         <Text>Company Name</Text>
263         <Input type="text" value={companyName} onChange={handleCompanyNameChange}></Input>
264         <Text>Email</Text>
265         <Input type="text" value={email} onChange={handleEmailChange}></Input>
266         <Text>Password</Text>
267         <Input type="password" value={password} onChange={handlePasswordChange}></Input>
268       </Form>
269       <Text>Sign In</Text>
270     </View>
271   );
272 }
```



LA SECURITE

Security(react-native)

La sécurité est souvent négligée lors de la création d'applications. Il est vrai qu'il est impossible de créer un logiciel complètement impénétrable - nous n'avons pas encore inventé un verrou complètement impénétrable (les coffres-forts des banques, après tout, sont toujours cambriolés). Cependant, la probabilité d'être victime d'une attaque malveillante ou d'être exposé à une faille de sécurité est inversement proportionnelle à l'effort que vous êtes prêt à déployer pour protéger votre application contre une telle éventualité. Bien qu'un cadenas ordinaire soit crochetable, il est toujours beaucoup plus difficile à franchir qu'un crochet d'armoire !

Storing Sensitive Info

Ne stockez jamais de clés API sensibles dans le code de votre application. Tout ce qui est inclus dans votre code peut être consulté en texte brut par toute personne inspectant l'ensemble d'applications. Des outils tels que react-native-dotenv et react-native-config sont parfaits pour ajouter des variables spécifiques à l'environnement telles que les points de terminaison d'API, mais ils ne doivent pas être confondus avec les variables d'environnement côté serveur, qui peuvent souvent contenir des secrets et des clés d'API.

Async Storage

Async Storage est un module géré par la communauté pour React Native qui fournit un magasin clé-valeur asynchrone et non chiffré. Le stockage asynchrone n'est pas partagé entre les applications : chaque application possède son propre environnement sandbox et n'a pas accès aux données des autres applications.

Network Security

Vos API doivent toujours utiliser le cryptage SSL. Le cryptage SSL protège contre la lecture des données demandées en texte brut entre le moment où elles quittent le serveur et avant qu'elles n'atteignent le client. Vous saurez que le point de terminaison est sécurisé, car il commence par https:// au lieu de http://.

Security (Symfony-Api-plateform)

composer require symfony/security-bundle

User(L'utilisateur)

Toute section sécurisée de votre application nécessite un certain concept d'utilisateur. Le fournisseur d'utilisateurs charge les utilisateurs à partir de n'importe quel stockage (par exemple, la base de données) sur la base d'un "identifiant d'utilisateur" (par exemple, l'adresse e-mail de l'utilisateur)

Les autorisations dans Symfony sont toujours liées à un objet utilisateur. Si vous avez besoin de sécuriser (des parties de) votre application, vous devez créer une classe d'utilisateurs. Il s'agit d'une classe qui implémente `UserInterface`. Il s'agit souvent d'une entité Doctrine, mais vous pouvez également utiliser une classe d'utilisateurs dédiée à la sécurité.

Registering the User: Hashing Password

De nombreuses applications nécessitent qu'un utilisateur se connecte avec un mot de passe. Pour ces applications, le `SecurityBundle` fournit des fonctionnalités de hachage et de vérification de mot de passe

The Firewall & Authenticating Users (firewalls)

Le pare-feu est au cœur de la sécurisation de votre application. Chaque demande dans le pare-feu est vérifiée si elle nécessite un utilisateur authentifié. Le pare-feu se charge également d'authentifier cet utilisateur (par exemple à l'aide d'un formulaire de connexion) ;

La section pare-feu de config/packages/security.yaml est la section la plus importante. Un "pare-feu" est votre système d'authentification : le pare-feu définit quelles parties de votre application sont sécurisées et comment vos utilisateurs pourront

s'authentifier (par exemple, formulaire de connexion, jeton API, etc

Access Control (Authorization) (access_control)

À l'aide du contrôle d'accès et du vérificateur d'autorisation, vous contrôlez les autorisations requises pour effectuer une action spécifique ou visiter une URL spécifique.

Outils utilisés

Draw oï : Organiser le user story par rapport à la priorité et à l'ordre d'importance.

Trello : Organiser et faire une estimation de compétences pour chaque tâche avec points relatifs à la complexité.

Figma : Création de notre maquette (wireframe, maquette et prototyping).

React-native : C'est un framework d'applications mobiles open source créé par Facebook. Il est utilisé pour développer des applications pour Android, iOS et UWP en permettant aux développeurs d'utiliser React avec les fonctionnalités natives de ces plateformes,

Expo : C'est un logiciel de développement plateforme pour développer une application universelle qui fonctionne sur Ios et Andriod en utilisant javascript framework.

Git : Plateforme de collaboration.

Xcode : Inclut tout ce dont les développeurs ont besoin

pour créer de superbes applications pour Mac, iPhone, iPad, Apple TV et Apple Watch. Xcode fournit aux développeurs un flux de travail unifié pour la conception, le codage, les tests et le débogage de l'interface utilisateur. L'IDE Xcode combiné au langage de programmation Swift rend le développement d'applications facile et amusant.



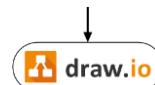
Xcode



Expo



Git



Conclusion

Pour conclure, je dirais que ce projet m'a fait prendre conscience des responsabilités que l'on peut avoir dans la réalisation d'un projet, ainsi que les difficultés à déceler les anomalies, et surtout à appréhender les non-conformités qui découlent de celles-ci.

L'expérience acquise au fil de ma formation pour devenir développeur web, mon implication personnelle et la recherche éternelle de connaissances m'ont été bénéfiques, et je pense avoir pris les bonnes décisions au cours des tâches qui m'étaient attribuées ainsi que dans mon implication dans celles-ci.