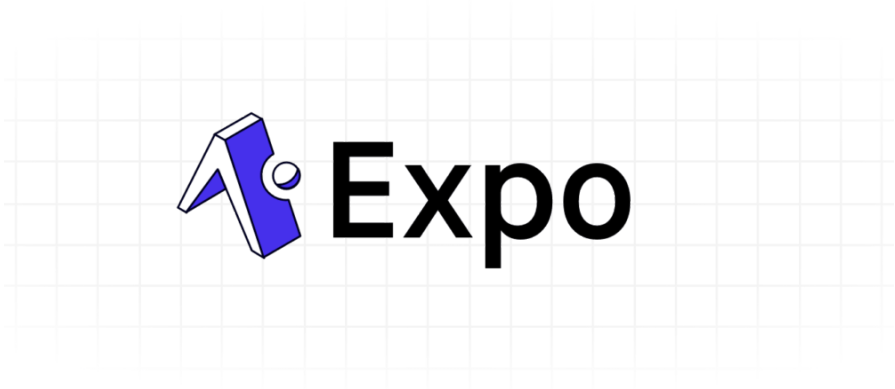
Dossier de projet Professionnel:

Application Mobile avec

React/Expo



Présenté par Julia Denivet

Sommaire

(a faire en dernier)

1.Introduction

a. Résumé du dossier du Projet Professionnel

Ce dossier présente l'application web mobile que j'ai pu concevoir et développer en collaboration avec mon camarade Luc au sein de mon cursus à l'école la Plateforme_. Ce dossier à pour but de présenter mon application web mobile, mon alternance au sein de Hopps Groups ainsi que quelques projets divers.

2.Liste des compétences couvertes par le projet Professionnel

Pour l'application mobile React/CodeIgniter couvre les compétences suivantes :

Pour l'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
- ❏ Développer la partie front-end d'une interface utilisateur web
- ❏ Développer la partie back-end d'une interface utilisateur web

Pour l'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

Pour l'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 à 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

3.Présentation

a. Présentation personnelle

Je m'appelle Julia Denivet, j'ai 21 ans et j'habite à Marseille dans les Bouches du Rhône. J'ai obtenu l'an dernier le titre professionnel Développeur web et web mobile.

Je souhaite maintenant devenir Experte développeuse web et mobile, c'est pour cela que j'ai intégré la formation Coding School à l'école La Plateforme_ et par la suite je compte intégrer un master Expert développeur web.

b. Présentation de l'école LaPlateforme_

L'école LaPlatefome_ est un centre de formation situé à Marseille. La Plateforme_ propose diverses formations autour du digital (développement web, cyber-sécurité, intelligence artificielle). En Septembre 2019, en partenariat avec le Top20, elle a mis en place une formation de développement web gratuite appelée Coding School, cette formation dure 2 ans.

c. Project summary, day release vocational training and mobile app

c.1 Alternance avec Hopps Groups

J'ai réalisé mon alternance avec Hopps Groups. C'est une entreprise faisant de l'e-logistique et du courrier. Hopps est le premier opérateur postal privé en France pour les professionnels et particuliers. Ce sont aussi des spécialistes internationaux de la livraison et de la logistique e-commerce. Finalement, ce sont les leaders privés des solutions de distribution du média courrier.

Durant cette alternance, j'ai pu apprendre de nouvelles technologies que je n'ai pas travaillé à l'école. J'ai commencé à travailler sur le framework Angular (plus communément appelé Angular 2+ ou Angular v2 et plus). C'est un open source qui utilise le langage TypeScript créé par la team Angular de Google et par une communauté d'individus et d'entreprises. Angular est une refonte du même groupe qui a développé AngularJS. Aussi, au cours de mon alternance je voulais faire du développement back-end donc j'ai commencé à travailler avec C# (prononcé "C Sharp") avec l'utilisation de Doc.NET. C# est un langage de programmation développé par Microsoft. C# a commencé à apparaître en 2002 avec la version 1.0 de Microsoft .Net framework. Depuis ça, C# est passé par plusieurs révisions qui correspondent avec chaque mise à jour de .NET.

De nos jours, C# est l'un des langages informatiques les plus populaires pour créer des logiciels Windows et des applications web. Dans mon cas, j'ai appris la langage C# pour créer des API pour le site extranet de Hopps Group.

c.2 project summary

To begin with, during my school days, I did the conception and development of a mobile app. This app is a shop for my teammates Luc. Luc is a photographer and he is passionate about that. Before that we decided to create this app, he only had instagram to post his shootings

.

The problem was the following : Luc needed more visibility, more photo models and maybe something that could help him to buy all he needed for his passion.

Also, we chose to do a shop which sells some pictures from shootings that Luc did before. This app is an opportunity for Luc to gain visibility for his skills and passion. On this app, we will see some architecture, landscape and photo models.

In addition to that, the app enables you to purchase his photoshootings and the money raised is spent to enhance his work equipment in order to always get the highest quality of photoshootings.

Furthermore, as a technical part, we decided to do an API RESTFUL for the back-end part and do a mobile application for the front-end part. We developed our API with codeigniter which uses the language PHP. Finally, the front-end was made with React Native which uses Javascript (XML).

In Fine, if by any chance you are interested in Luc's work you can find it on the app Luc's contact details to call him or send him a mail.

4.Cahier des charges

Ce cahier des charges a été élaboré avec Luc avec qui j'ai développé ce projet.

- ☐ Charte Graphique : site minmaliste avec un thème plutôt noir et blanc.
- ☐ Une boutique de photos avec plusieurs catégories : Architecture, Astrophoto, paysage et Shooting photo
- ☐ Un panier avec un système de paiement spécifique
- ☐ Une page de Connexion/Inscription avec Token du côté API
- ☐ Un système de favoris pour voir les photos les plus vendues
- ☐ Création d'un système d'administration accessible uniquement par mot de passe et id droits spécifiques. Possibilité de modifier, ajouter ou supprimer certaines photos, catégories, utilisateurs etc.
- ☐ Mise en place de pages standard (page d'accueil, header, contact..)

5. Spécification technique du projet

a.Technologies utilisées

L'application mobile la boutique de Luc est destinée à être utilisée sur téléphone. L'application a été développée avec une API REST codée à l'aide du framework PHP CodeIgniter. La partie front a été développée avec un framework Javascript ReactNative. Pour pouvoir visualiser le projet nous avons utilisé expo qui permet de pouvoir directement observer l'application sur le téléphone.

a.1 API REST et CodeIgniter

L'acronyme API signifie "Application Programming Interface" en Anglais et peut être traduit simplement par "Interface de Programmation d'Application". Pour rester simple, il s'agit de créer un accès à une application et permettre de venir y consommer des données ou des fonctionnalités. Ainsi, grâce à ça avec une autre application ou un site, venir interroger l'application et utiliser ses données.

Pour développer cette API, j'ai utilisé CodeIgniter. Codeigniter est un framework PHP simple, léger et performant, sur le modèle de développement MVC. CodeIgniter reprend un design pattern de composition : HMVC (Hierarchical Model

Vue Controller). Ce modèle dérive du MVC (Model View Controller) qui prône la séparation des préoccupations (separation of concerns).

Si le modèle MVC permet une programmation élégante et plus maintenable/évolutive de part la séparation claire entre l'acquisition de données (le M de Model), la présentation de ces données (le V de Vue) et la gestion des demandes des utilisateurs (le C de Controller). Le modèle HMVC y ajoute une composante hiérarchique pour encore plus de modalités.

Il est possible d'utiliser des librairies/helpers fournis entre autres par CodeIgniter permettant d'enrichir ce modèle.

a.2 React Native et Expo

ReactNative est un framework développé par les équipes de Facebook à partir de 2015 dans le but d'accélérer le développement et la maintenance de leurs applications mobiles. Cette technologie multi-platform a gagné une immense popularité depuis son lancement et c'est l'un des projets les plus populaires sur GitHub. C'est un framework qui utilise le langage de programmation Javascript et les librairies React créées par Facebook pour développer des interfaces web.

Pour pouvoir visualiser ce qui a été fait avec React nous avons utilisé expo. Expo est un framework et une plateforme qui simplifient la création et le déploiement d'applications mobiles avec React Native. 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.

- ☐ **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).

Pour ce qui est du style nous avons utilisé principalement du CSS et un peu de Bootstrap.

Concernant l'IDE j'ai développé mon projet avec Visual Studio Code et j'ai aussi utilisé Postman pour pouvoir effectuer mes tests d'api.

b. Versionning et stockage

Pour le stockage j'utilise Git. Nous avons travaillé sur plusieurs branches : 1 branche Master, 1 branche pour Luc et une branche pour Julia.

c. Workflow

Lors de ce projet, nous avons travaillé chacun de notre côté . Régulièrement, après certaines modifications nous "commitions" et "push" le code sur GitHub. Souvent, j'ai pull pour récupérer la base de données qui a été régulièrement modifiée au cours du projet.

d.Répartition du travail

Pour l'application mobile en react native, nous avons utilisé trello pour la répartition des tâches avec des deadlines pour les différentes tâches que l'on a effectué que ce soit du côté API que du côté front. Ce projet a commencé en décembre 2020 et devrait être terminé très prochainement.

e. Documentation utilisé

Lors de ce projet, internet m'a beaucoup aidé. Principalement, j'ai consulté les sites de React, CodeIgniter et Expo ainsi que les sites parlant du JWT Token.

Exemple des sites de recherche utilisés :

- ☐ https://reactnative.dev/
- ☐ https://expo.io/
- ☐ https://www.codeigniter.com/
- ☐ https://www.ionos.fr/digitalguide/sites-internet/developpement-web/json-web-token-jwt/

6. Réalisation

a.Réaliser la partie front-end d'une application

a.1 Arborescence du site

Lorsque l'on arrive sur l'application Boutique de Luc, on arrive sur la page d'accueil qui est une page assez minimaliste qui comprend un background correspondant à une photo prise par Luc à la Joliette ainsi que les photos favorites de la Boutique.

G

Grâce à la navigation, l'utilisateur peut facilement changer de page. Pour commencer, il peut s'inscrire et se connecter à l'aide d'un formulaire.
Lorsqu'il est enfin connecté, il peut remplir son panier avec les photos qu'il souhaite acheter.
Pour trouver ces photos, il faut se rendre sur le module boutique qui contient toutes les photos à vendre sur l'application et nous pouvons les trier par catégorie.

Dès que l'utilisateur a ajouté ce qu'il voulait à son panier, il peut se rendre sur la page panier ou il pourra voir le visuel de son panier ainsi que le total de celui-ci. Il pourra à terme valider son panier, choisir une adresse de facturation ainsi qu'une adresse de livraison et choisir un moyen de paiement.

Quand il le souhaitera, l'utilisateur peut modifier son profil grâce au module EditUser. Dans cette page il pourra aussi visualiser l'historique de ses commandes.

L'utilisateur pourra retrouver les contacts du photographe dans la navigation de l'application.
Pour terminer l'arborescence de notre application, le module administrateur est uniquement accessible par connexion sécurisée avec vérification de l'id droit. Ce module va permettre à l'admin de pouvoir :

- Modifier/Ajouter/Supprimer un utilisateur
- Modifier/Ajouter/Supprimer un nouveau produit
- Modifier/Ajouter/Supprimer une nouvelle catégorie

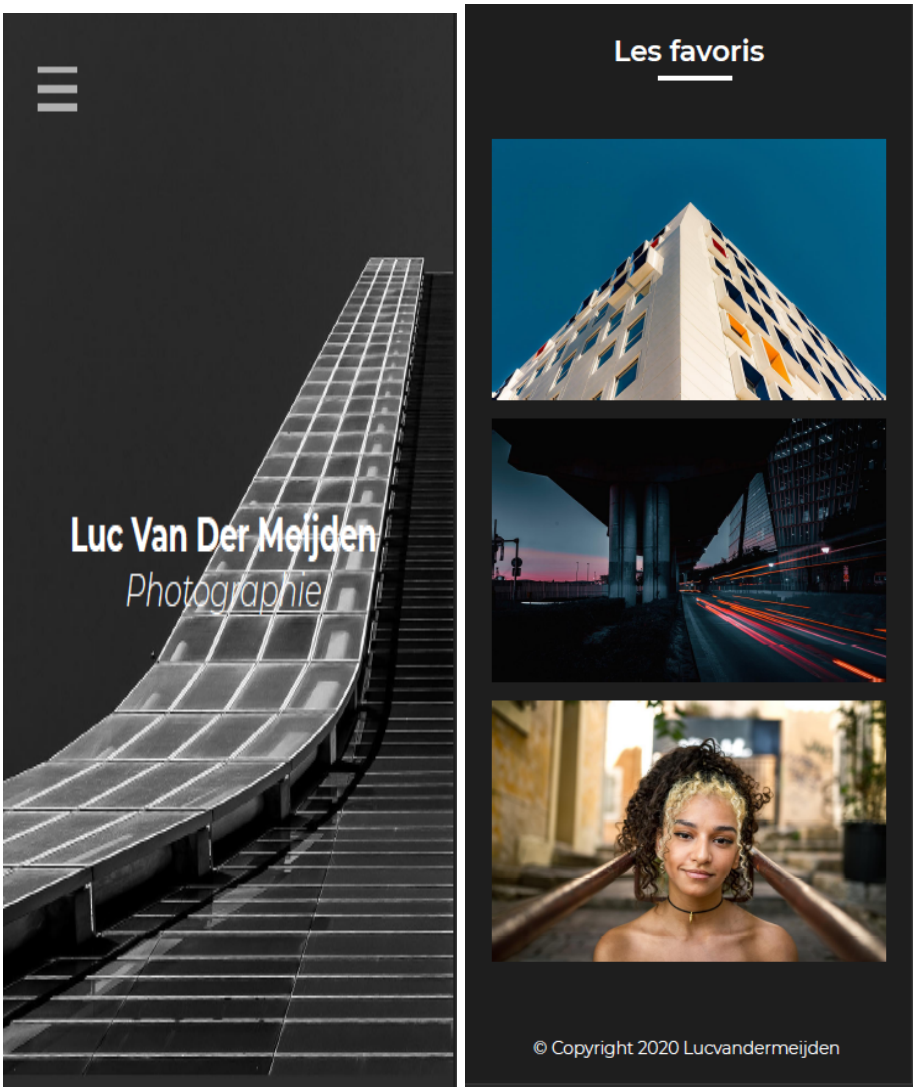
a.2 Maquettage du site

Grâce à nos réunions quasi quotidiennes nous avons pu élaborer un maquette complète de notre site. Le souhait de Luc était à terme d'avoir une boutique en ligne de ses photos.

Pour faire la maquette nous avons utilisé Sketch. Cette application est un logiciel "

qui sert principalement à faire des maquettes.

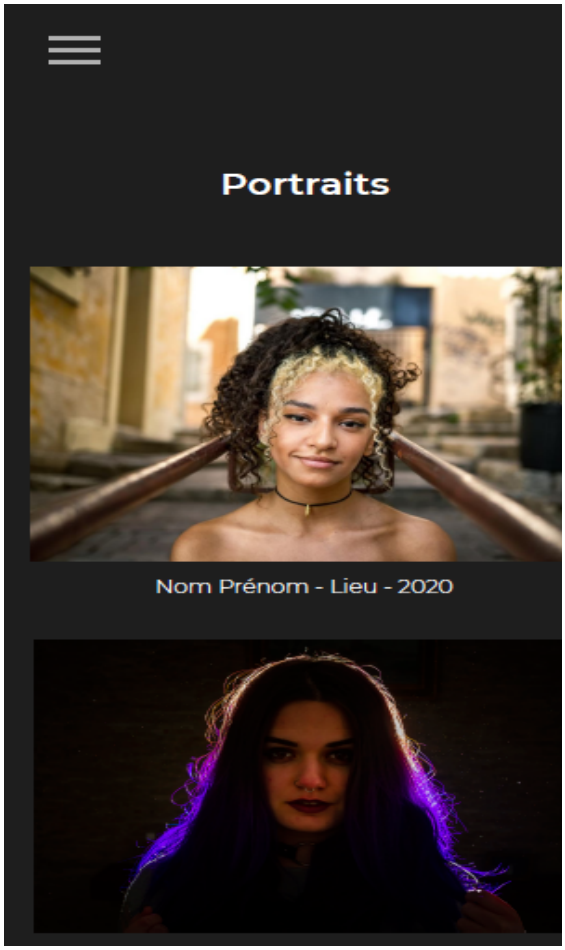
Pour commencer, nous avons fait la maquette de la page d'accueil. Cette page est la première page que l'on voit lorsque l'on arrive sur l'application. Le thème graphique est assez minimaliste. Nous avons un background avec une architecture ainsi que le visuel des favoris.



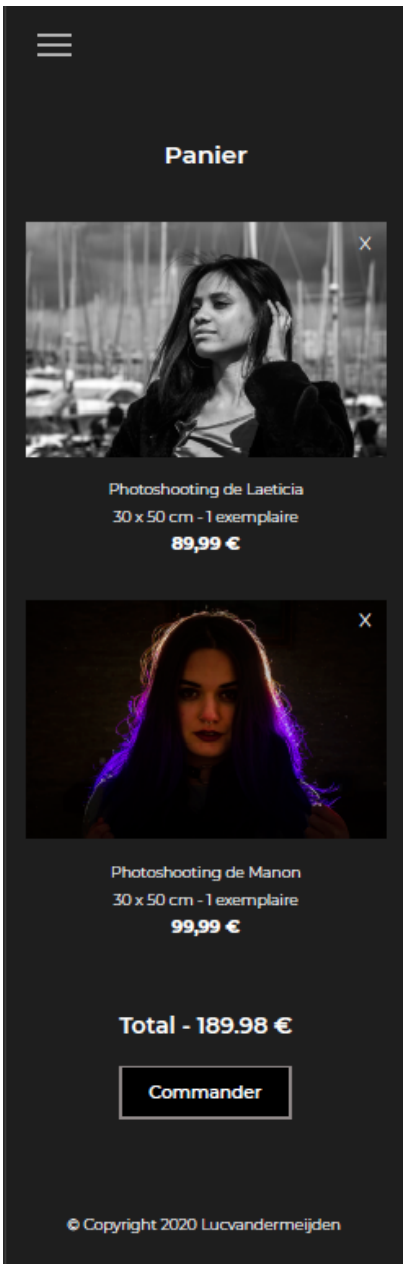
Pour que l'utilisateur puisse ajouter des éléments à son panier, il a besoin de se créer un compte et de se connecter. Pour la maquette, nous avons décidé de faire deux pages assez similaires avec des formulaires simples. Pour la connexion, cela se fait avec email et mot de passe. L'inscription, elle, est plus complète.

Pour s'inscrire, il faut renseigner : son mail, son mot de passe et la confirmation de son mot de passe.

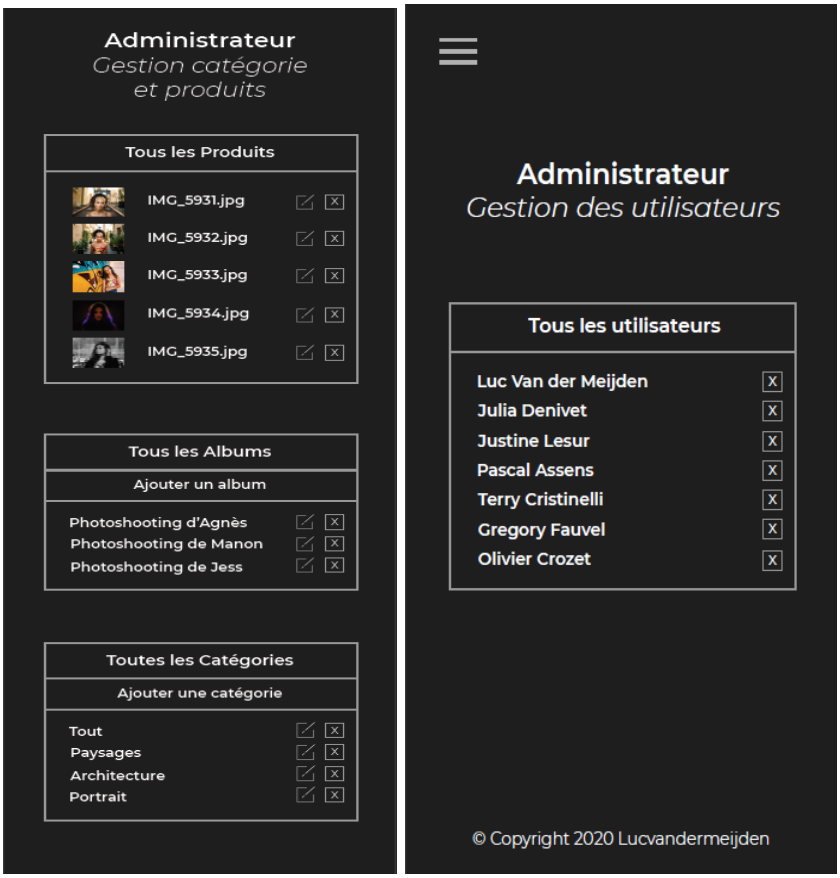
La page Boutique est séparée en plusieurs pages . Chaque page correspond à une catégorie : Photoshooting, architecture et paysage. On y voit la photo ainsi que sa description (nom – lieu – taille – prix) et une icône d'ajout au panier.



Le panier comporte les produits ainsi que leur description et leur prix. En bas de page, on retrouve le total du panier suivi d'un bouton commande qui amènera à terme sur une page pour valider la commande avec le moyen de paiement.



Nous avons terminé la maquette par la partie administration. Le module d'administration est divisé en deux parties : la partie utilisateur et la partie boutique. La partie utilisateur comporte un tableau avec tous les utilisateurs inscrits en base de données avec des icônes permettant d'ouvrir une pop-up avec un formulaire de modification d'utilisateur. La partie boutique permet d'ajouter, modifier et supprimer une catégorie ou un produit avec ouverture d'une pop-up avec un formulaire de modification.



a.4 La NavBar

Pour la navigation, nous avons décidé de travailler avec le Drawner Navigation de React Native. Le drawner navigation est un composant permettant d'avoir un menu s'ouvrant et fermant par un geste (slide vers la droite/gauche).

Avant de commencer, j'ai installer react navigation et react-navigation/drawer avec npm avec les commandes suivantes :

```
npm install @react-navigation/native
```

```
npm install @react-navigation/drawer
```

Pour le code, dans mon composant app.je, je crée des fonctions pour chaque page et je fais ensuite un appel au composant correspondant à la page. Par exemple, je crée une fonction Home() qui va faire appel au composant accueil. Je n'oublie pas de faire mes imports pour les composants. Je commence ensuite à créer la structure de ma navigation. J'utilise la balise <NavigationContainer> pour commencer ma navigation. J'ouvre ensuite une balise <Drawner.Navigator> en lui donnant ma route initial "home" ce qui signifie que la premier composant appelé à l'ouverture de l'application est la page d'accueil.

Ensuite, j'utilise la balise <Drawner.screen> qui va me permettre de lier ma barre de navigation avec mes composants. Cette balise comporte plusieurs paramètres :

- ☐ son nom (exemple : Connexion)
- ☐ le fonction que j'ai crée juste avant (exemple : {ConnexionScreen})

La Barre de Navigation fonctionne correctement et facilite l'accès aux autres pages.

```
function HomeScreen({ navigation }) {
  return (
    <View>
     <Accueil/>
    </View>
  );
}
```

```
<Drawer.Screen name="Accueil" component={HomeScreen} />
<Drawer.Screen name="Connexion" component={ConnexionScreen} />
<Drawer.Screen name="Inscription" component={SingUpScreen} />
<Drawer.Screen name="Architecture" component={ArchitectureScreen} />
<Drawer.Screen name="Paysages" component={PaysagesScreen} />
```

a.5 Composants et structure

Boutique + formulaire balise + style + cycle de vie d'un composant

a.6 Axios + le token

b.Réaliser la partie Back-end d'une application, API

b.1 Conception et création de la base de données

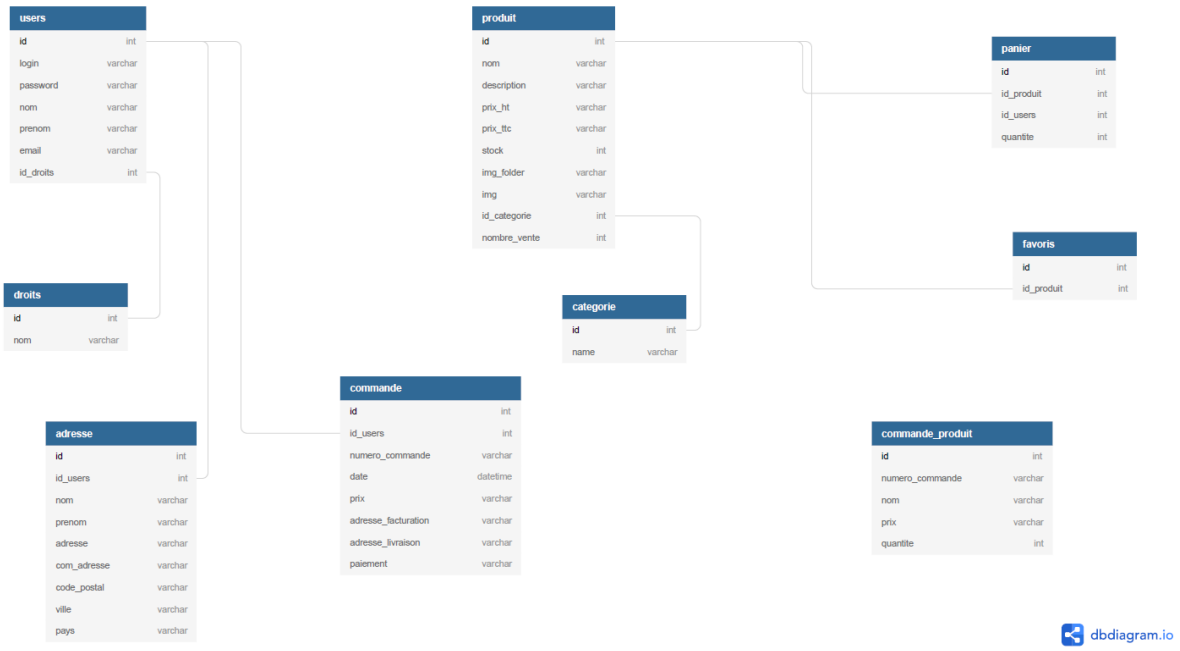
Pour la boutique de Luc, j'ai élaboré et créé la base de données. Vu que nous sommes sur une boutique, il y a besoin d'une authentification pour pouvoir acheter les produits et cela nécessite la création d'une table user.
Nous avons aussi besoin d'une table produits pour la boutique ainsi qu'une table catégorie qui sera liée à la table produits.

Dans une boutique, nous avons besoin d'un panier pour que l'utilisateur puisse avoir un récapitulatif des produits qu'il ajoute ainsi que sa quantité et le prix total de son panier. Pour ça, nous avons créé deux tables : une table panier qui lorsqu' un produit sera ajouté au panier récupérera id_produit ainsi que sa quantité et l'id_users qui permettra d'identifier à qui appartient le panier en cours.

Après la validation du panier, on va envoyer les produits, la quantité et le prix à la table commande_produit qui a en plus un numéro de commande. Si un utilisateur fait une commande avec plusieurs produits, ces produits auront le même numéro de commande. Cette commande finit dans la table commande qui prend en compte l'utilisateur connecté, les produits qu'il a achetés, le paiement, le prix ainsi que les adresses de facturation et de livraison. Ces adresses pourront être récupérées qui gardent en mémoire les adresses enregistrées par l'utilisateur.

Pour finir, nous avons décidé d'avoir une table favoris qui récupère uniquement l'id du produit favoris. Pour la connexion, nous avons créé une table de droit pour l'accès à la page d'administration. Cette table à deux colonnes : id (1 ou 1337) et un nom.

Pour la maquette de la base de données, j'ai utilisé le logiciel dbdiagram.io



Et voici l'outil permettant de faire ce diagramme :

```
//// -- Tables and References

// Creating tables
Table users as U {
  id int [pk, increment] // auto-inc
  login varchar
  password varchar
  nom varchar
  prenom varchar
  email varchar
  id_droits int

}

Table droits as D {
  id int [pk, increment]
  nom varchar
}

Table produit as P {
  id int [pk, increment] // auto-inc
  nom varchar
  description varchar
  prix_ht varchar
  prix_ttc varchar
  stock int
  img_folder varchar
  img varchar
  id_categorie int
  nombre_vente int

}

Table panier as S{
  id int [pk, increment] // auto-inc
  id_produit int
  id_users int
  quantite int

}

Table favoris as F {
  id int [pk, increment]
  id_produit int
}
```

Voilà ce que cela donne dans PHPMyAdmin :

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	id	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT	Modifier Supprimer Plus
☐	2	id_users	int(11)			Non	*Aucun(e)*			Modifier Supprimer Plus
☐	3	numero_commande	varchar(255)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	4	date	datetime			Non	*Aucun(e)*			Modifier Supprimer Plus
☐	5	prix	varchar(255)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	6	adresse_facturation	varchar(255)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	7	adresse_livraison	varchar(255)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	8	paiement	varchar(255)	utf8_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus

Lorsque la base de données est enfin créée nous pouvons commencer à créer notre API.

b.2 L'API RESTful

L' APi RESTful contrairement à l'API SOAP est une architecture. REST utilise des méthodes HTTP pour récupérer et publier des données entre un périphérique client et un serveur. Les clients de l'API utilisent des appels HTTP pour demander une ressource (une méthode GET) ou envoyer des données au serveur (une méthode POST), ou l'une des autres méthodes HTTP prises en charge par l'API. GET et POST sont les méthodes les plus fréquemment utilisées, mais d'autres méthodes comme HEAD, PUT, PATCH, DELETE, CONNECT, OPTIONS et TRACE peuvent également être prises en charge.

Pour le création de cette API, nous avons décidé de travailler avec le framework COdeigniter qui est un framework PHP utilisant une architecture HMVC en couches.

Pour chaque entité de l'API, nous avons fait un CRUD : Create, Read, Update et Delete. Nous allons prendre pour exemple l'entité user.

b.3 L'entité User

L'entité User est l'une des entités les plus importantes de la Boutique. Elle permet l'affichage, la création, la modification de l'utilisateur. L'avantage de l'utilisation de CodeIgniter permet d'avoir plusieurs couches : le Model qui permet de faire les appels à la base de donnés, le controller qui permet de manipuler les données envoyées par le model c'est aussi le routeur de framework qui va faire le lien entre l'url et la classe/méthode. Dans notre API, nous avons decider d'utiliser des fat models et des thins controller. Cela signifie que ce sont nos models qui contiennent les logiques métiers. Et enfin la Vue qui est responsable de la présentation des données. Dans notre cas, la vue est faite avec ReactNative.

Dans cette partie nous ne parlerons pas encore du JWT Token...

b.3.1 La méthode GetUser()

```php
public function index()
{
    try
    {
        $data=$this->Models_Users->getUsers();
    }
    catch (Exception $e)
    {
        echo 'Exception reçue : ',  $e->getMessage(), "\n";
    }
}
```

Controller user

```php
public function getUsers()
{
    $received_Token = $this->input->request_headers('Authorization');
    try
    {
        $jwtData = $this->objOfJwt->DecodeToken($received_Token['Token']);
        $query = $this->db->query("SELECT * FROM users WHERE id='".$jwtData['id']."'");
        $data = $query->row_array();
        echo json_encode($data);
    }
    catch (Exception $e)
    {
        http_response_code('401');
        echo json_encode(array( "status" => false, "message" => $e->getMessage()));exit;
    }
}
```

Model Users

Ici, dans l'affichage de l'utilisateur nous créons la méthode GetUsers() dans le model et Index() dans le controller.

Du côté du model, nous utilisons le query pour exécuter la requête "SELECT * FROM users WHERE id = "' .$jwtDate['id'].'"" qui va demander à la base de données boutique de tout sélectionné dans la table users lorsque l'id correspond à l'id de l'utilisateur demandé. On utilise ensuite le row_array() (va devenir result_array()). pour pouvoir récupérer les résultats de la requête qu'on a fait à la base de données. Avant d'être recuperer par le controller, je n'oublie pas d'utiliser la méthode json-encode sur mon résultat $data qui va me retourner la représentation json de $data.
Pour finir dans le model, j'utilise un catch exception qui dit que si l'utilisateur n'est pas le nom je renvoie une erreur 401 qui signifie que la personne n'est pas autorisé car il n'a pas été identifié.

Du côté controller, on fait appel à la méthode du model qui récupère mes résultats et qui les stocke dans une variable $data. Par la suite, quand dans mon react je vais faire appel avec axios à la fonction getusers() j'afficherais la liste des utilisateurs. Passons maintenant à la création de l'utilisateur :

b.3.2 La méthode create()

```php
public function create($data)
{
    $data['password'] = password_hash($data['password'], PASSWORD_DEFAULT);
    echo json_encode($data);
    $this->db->insert('users', $data);
}
```

Model User

```php
public function create()
{
    try
    {
        $data = array(
            'lastname' => $this->input->post('lastname'),
            'firstname' => $this->input->post('firstname'),
            'email' => $this->input->post('email'),
            'password' => $this->input->post('password'),
            'id_droits' => '1'
        );

        $this->Models_Users->create($data);
    }
    catch (Exception $e)
    {
        echo 'Exception reçue : ',  $e->getMessage(), "\n";
    }
}
```

Controller User

La méthode create() est la méthode qui permet la création de l'utilisateur en base de données. Lors de la création, l'utilisateur doit renseigner : le nom, le prénom, un mail et un mot de passe. Nous allons donc créer deux méthodes create() : une pour le model et une pour le controller.

Pour le controller, nous avons besoin de créer un tableau pour tous les inputs du formulaire (nom, prénom, mail, mot de passe). Nous stockons ce tableau dans une variable $data qu'on envoi ensuite en paramètre à la méthode create() du model.

La méthode va ensuite récupérer le contenu de la variable data pour ensuite la manipuler et envoyer ça à la base de données. On va d'abord crypter le mot de passe avant de l'envoyer à la base de données. Ensuite, comme pour l'affichage on utilise la méthode json_encode sur les données que l'on veut insérer dans la base de données. On insère ensuite dans la table user la variable $data qui contient les informations du nouvel utilisateur. Après ça, la création de l'utilisateur est fonctionnelle.

b.3.3 La méthode modify()

```php
public function modify()
{
    try
    {
        $data = array(
            'lastname' => $this->input->post('lastname'),
            'firstname' => $this->input->post('firstname'),
            'email' => $this->input->post('email'),
            'password' => $this->input->post('password'),
            'id_droits' => $this->input->post('id_droits')
        );

        $this->Models_Users->modify($data);
    }
    catch (Exception $e)
    {
        echo 'Exception reçue : ',  $e->getMessage(), "\n";
    }
}
```

Controller User

```php
public function modify($data)
{
    $received_Token = $this->input->request_headers('Authorization');
    try
    {
        $jwtData = $this->objOfJwt->DecodeToken($received_Token['Token']);
        $query = $this->db->query("SELECT * FROM users WHERE id='".$jwtData['id']."'");
        $data2 = $query->row_array();

        if($data['password'] != $data2['password'])
        {
            $data['password'] = password_hash($data['password'], PASSWORD_DEFAULT);
            $this->db->where("id=", $data2['id']);
            $this->db->update('users', $data);
        }
        else
        {
            $this->db->where("id=", $data2['id']);
            $this->db->update('users', $data);
        }
        echo json_encode($data);
    }
    catch (Exception $e)
    {
        http_response_code('401');
        echo json_encode(array( "status" => false, "message" => $e->getMessage()));exit;
    }
}
```

Model User

Pour le controller, on utilise la même méthode que pour le create(). Lorsque notre tableau de données est envoyé au model il y a deux cas. Le début de la requête ne change pas. Je sélectionne d'abord toutes mes informations de la table user ou l'id de la table correspond à l'id de l'utilisateur. Je vais ensuite vérifier si l'utilisateur a demandé la modification de son mot de passe si c'est le cas, on crypte le nouveau mot de passe et on modifie les informations de l'utilisateur en vérifiant que les id correspondent bien. Dans l'autre cas, on a pas besoin de crypter le mot de passe on envoie seulement les éléments qui ont été modifiés en vérifiant bien que l'id reçu de l'utilisateur et l'utilisateur dans la base de données soient les mêmes.

b.3.4 La méthode delete()

```php
public function delete($data)
{
    $received_Token = $this->input->request_headers('Authorization');
    try
    {
        $jwtData = $this->objOfJwt->DecodeToken($received_Token['Token']);
        if($jwtData['id_droits'] == "1336")
        {
            echo json_encode($data);
            $this->db->where("id=", $data['id']);
            $this->db->delete('users', $data);
        }
    }
    catch (Exception $e)
    {
        http_response_code('401');
        echo json_encode(array( "status" => false, "message" => $e->getMessage()));exit;
    }
}
```

Model User Admin

```php
public function delete()
{
    try
    {
        $data = array(
            'id' => $this->input->post('id')
        );

        $this->Models_Users_Admin->delete($data);
    }
    catch (Exception $e)
    {
        echo 'Exception reçue : ',  $e->getMessage(), "\n";
    }
}
```

Controller User Admin

La méthode delete() est présente uniquement dans la partie administration pour le moment.

Dans le controller, on stocke un tableau dans la variable $data, elle contient un tableau avec l'id de l'utilisateur qu'on veut supprimer. On envoie ensuite la variable $data en paramètre à la méthode delete() du model.

Dans le model, on vérifie tout d'abord si la personne qui a fait la demande de suppression a un id_droit egal à 1336. Si c'est le cas, nous pouvons continuer la méthode de suppresion. On affecte la méthode json_encore() à la variable $data, on recupère la ligne avec l'id user qui correspond à l'id qu'on veut supprimer et on supprime la ligne avec le $this->db->delete('users', $data).

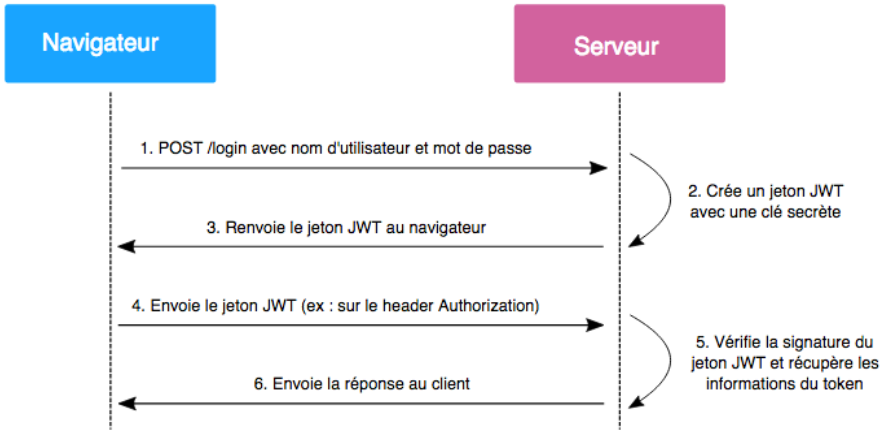
Test requête delete() image

b.4 JWT Token

Les « JSON Web Token » ou JWT sont des jetons générés par un serveur lors de l'authentification d'un utilisateur sur une application Web, et qui sont ensuite transmis au client. Ils sont renvoyés avec chaque requête HTTP au serveur, ce qui lui permet d'identifier l'utilisateur.

Pour ce faire, les informations contenues dans le jeton sont signées à l'aide d'une clé privée détenue par le serveur. Quand il recevra à nouveau le jeton, le serveur n'aura qu'à comparer la signature envoyée par le client et celle qu'il aura générée avec sa propre clé privée et à comparer les résultats. Si les signatures sont identiques, le jeton est valide.

Schéma explicatif de comment marche le token :



Pour revenir à notre projet, nous appelons pour la première fois le jwt token à la connexion. Quand un utilisateur se connecte, on fait

7.Présentation du jeu d'essai

_____Les deux fonctionnalités les plus importantes de l'application sont :

- La boutique séparée en plusieurs catégories permet à l'utilisateur de pouvoir choisir les produits qu'ils veulent acheter et aussi voir les descriptions des produits.
- Le panier qui permet à l'utilisateur de voir les produits qu'il a ajoutés, en quelle quantité et le prix total de son panier.

a.La Boutique

(visuel de la boutique)

b.Le Panier

(visuel du panier)

8. Vulnérabilités de sécurité

9. Extrait du site pour la recherche

La méthode `componentDidMount()` est exécutée après que la sortie du composant a été injectée dans le DOM. C'est un bon endroit pour mettre en place le minuteur :

```
componentDidMount() {
  this.timerID = setInterval(
    () => this.tick(),
    1000
  );
}
```

Notez qu'on a enregistré l'ID du minuteur directement sur `this` (`this.timerID`).

Alors que `this.props` est mis en place par React lui-même et que `this.state` a un sens bien spécial, vous pouvez très bien ajouter manuellement d'autres champs sur la classe si vous avez besoin de stocker quelque chose qui ne participe pas au flux de données (comme un ID de minuteur).

Lors du développement de l'application avec ReactNative, j'ai créé et utilisé des composants pour mes pages. Pour cela, j'ai effectué des recherches pour comprendre comment fonctionne et quelles sont les étapes lors de la génération de mon composant. Je me suis surtout penché sur la méthode componentDidMout() car pour ma partie admin utilisateur je voulais que le tableau s'affiche directement après le chargement de mon composant. C'est la méthode appelée lorsque l'on veut charger des données depuis un point d'accès distant.

10. Synthèse du projet

satisfactions personnelles + satisfaction du projet + conclusion alternance + quoi faire après le diplôme (master+alterance)