Android – Laravel – VueJS

Projet d'intégration

Kévin Oliva – LP NUM

Année universitaire 2020/2021

Table des matières

1. Déploiement

Dans un premier temps, nous allons récupérer notre projet sur GitHub.

Ouvrez un terminal de commande et exécutez la commande suivante :

```
git clone https://github.com/kevoliva/interventions.git
```

Soyez sûrs d'avoir Composer, Yarn et Android Studio installés sur votre machine.

1.1 Laravel

Rendez-vous dans le dossier interventions/api_interventions. C'est notre projet Laravel.

Créez le fichier « .env » à la racine du dossier, et insérez le code suivant :

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:tdX1GJoE7DJ6c42otDSHUkba44FWe48/1znzOLW0kk4=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=api_interventions
DB_USERNAME=?????
DB_PASSWORD=?????

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
```

```
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Remplacez les « **?????** » par l'identifiant et le mot de passe de votre base de données.

Dans cette dernière, créez une base de données nommée précisément « **api_interventions** ».

Exécutez maintenant les commandes suivantes :

```
composer install

php artisan migrate

php artisan serve
```

L'api est maintenant configurée.

1.2 VueJs

Rendez-vous dans le dossier « front » et exécutez simplement les commandes suivantes :

```
yarn

yarn dev
```

1.3 Android

Ouvrez Android Studio, et ouvrez simplement le projet nommé « Android ».

2. Développement

2.1 Laravel

Pour Laravel, nous avons dû réaliser une api, permettant de recevoir les données transmises par notre application Android afin de les envoyer à notre application VueJs.

Pour cela il a fallu réaliser un mini schéma de base de données :

Un utilisateur (technicien) peut réaliser plusieurs interventions.

Une intervention ne peut être réalisée que par un utilisateur (technicien).

Un utilisateur comprend les attributs suivants :

- identifiant
- nom
- email
- password
- created_at
- updated_at

Une intervention comprend les attributs suivants :

- identifiant
- user_id (clé étrangère)
- nomClient
- prenomClient
- adresseClient
- marqueChaudiere
- modeleChaudiere
- dateMiseEnService
- numeroSerie
- description
- tempsPasse
- created_at
- updated_at

Il a également fallu désigner des règles de validation afin de ne pas rentrer n'importe quoi comme valeurs. Elles se trouvent dans StoreUser et StoreIntervention.

Afin de pouvoir POST des données, il a fallu attribuer la valeur TRUE au booléen qui gère cette autorisation.

2.2 VueJs

Pour VueJs, nous devons récupérer les données de l'api. Toutes les méthodes GET et POST sont implémentées dans le fichier « store.js ». J'ai choisi de créer une méthode postUser, qui permettra à l'utilisateur du site (l'administrateur, probablement de patron de l'entreprise), de pouvoir créer des utilisateurs, qui pourront donc par la suite se connecter à l'application Android afin de remplir des fiches d'intervention.

J'ai choisi de placer des transitions entre les pages.

J'ai également développé des fonctions permettant de trier les techniciens/les interventions et/ou de les rechercher par nom. La pagination est aussi gérée.

2.3 Android

Pour Android, j'ai choisi de développer des choses assez simples, rapides, mais malheureusement pas assez sécurisées :

J'ai choisi d'utiliser une authentification (étant donné que l'administrateur peut créer des utilisateurs via VueJs). En revanche cette authentification utilise un GET et non un POST. Elle récupère tous les utilisateurs de l'API (dont le mot de passe, en clair…) et cherche si l'identifiant et le mot de passe saisis par l'utilisateur existent. Si oui, l'application se connecte, sinon non. Avec un peu plus de temps j'aurais probablement effectué un POST afin que ce soit Laravel qui gère l'authentification.

Lorsqu'un technicien saisit une intervention, elle est stockée dans la base de données d'Android. Grâce au bouton « Envoyer les interventions », ces dernières s'envoient à l'api grâce à une méthode POST (à condition d'avoir une connexion internet). Cette dernière reste toujours enregistrée dans la base d'Android.

J'ai fait ce choix car le technicien pourra par la suite consulter toutes les interventions qu'il aura réalisées (dans l'ordre chronologique). Lors de l'appui sur le bouton « Envoyer », seules les interventions n'ayant jamais été envoyée auparavant pourront s'envoyer à l'api (grâce à un booléen qui est seulement dans la base de données d'Android : « estEnvoye »).

3. Manuel d'utilisation

3.1 VueJs

L'application VueJs est très simple à utiliser. Il faut l'utiliser en premier car elle permet de créer des comptes pour les techniciens.

Il suffit de se rendre en haut à droite « Ajouter un technicien », puis de remplir un mini formulaire avec nom, prénom, email et mot de passe. L'utilisateur est ensuite créé, et pourra se connecter à Android.

Imaginons que cet utilisateur ait déjà renseigné et envoyé quelques interventions :

En arrivant sur la page d'accueil de l'application, nous avons accès à tous les techniciens. En cliquant sur un des techniciens (petit œil à droite), nous accédons à toutes les interventions réalisées par celui-ci. Il est également possible d'accéder à l'ensemble des interventions de tous les techniciens en cliquant sur « Toutes les interventions » en haut à droite de l'application.

Lorsque nous sommes face à la liste des techniciens ou face à la liste des interventions, nous avons la possibilité de filtrer et de trier ces derniers :

En cliquant sur une colonne, cela va trier le tableau en fonction de la colonne choisie (nom, date…).

La barre de recherche permet de saisir un nom (ATTENTION, la casse est respectée). Cela filtrera instantanément le tableau.

Enfin, nous avons une pagination afin de ne pas se retrouver avec un tableau trop grand.

3.2 Android

Dans un premier temps, nous arrivons sur une page de connexion. Le technicien doit saisir son adresse mail et son mot de passe (fournis par le patron de l'entreprise). Malheureusement (un soucis de fonction asynchrone ?), il faut appuyer une première fois sur le bouton connexion, puis une deuxième fois 2 secondes plus tard (le temps que l'api envoie les informations et qu'Android les traite) : problème à régler.

Ensuite, cliquez sur le bouton « Ajouter une intervention ». Remplissez le formulaire, puis enregistrez l'intervention.

Elle apparaît maintenant dans la liste des interventions (avec la spécification : « non envoyé »). Ensuite, le fait de cliquer sur « Envoyer les interventions » permettra à l'application de POST toutes les interventions jamais envoyées auparavant. Un petit message indiquera à l'utilisateur que les interventions sont bien envoyées, ou lui dira simplement qu'aucune nouvelle intervention n'est à ajouter.