AREA

Documentation technique

Nathan Hoche, Théo Bares,
Sevan Sarikaya & José Rodrigues

Epitech Toulouse 2021

Sommaire

Nathan Hoche, Théo Bares,
Sevan Sarikaya & José Rodrigues

1- Context du projet

Le projet AREA consiste à implémenter une suite logicielle composée de trois parties (serveur, client web et client mobile) afin de reproduire un projet similaire à l'IFTTT ou Zapier. Le serveur devra disposer de toutes les fonctionnalités demandées pour la réalisation du projet.
Le client web et mobile pourront ensuite interroger le serveur afin que l'utilisateur puisse réaliser des actions préalablement définies par celui-ci.

2- Matrice de décision :

2.1. web:

Framework	doc	Env	modularité	lib	Initial release	utilisé par	Popularité	Communauté
React	+++	Js	+++ (plus libre car c'est une lib)	npm	2013	Facebook, Uber, Netlix	top 1	+++
Angular	+++	Ts mais pb de compatibilité entre les versions	+++ (mieux encadré)	npm	2010	Google, Wix, Sony	2	+++
Vue	++	js	--	npm	2014	Alibaba, GitLab	3	++
Django	++	Python	(++ très lourd)	python	2005	instagram, pinterest	4	++
flask	+	Python	-	très peu	2010	peu utilisé	5	+

2.2. Mobile:

Vu que nous avons décidé de réaliser le site web en ReactJs, nous avons décidé de créer l'app mobile en React Native vu que les deux technologies se ressemblent fortement.

2.3. Serveur:

Lors du précédent projet nous avions décidé d'utiliser Express pour s'occuper du backend, du fait de sa simplicité d'utilisations, or nous avons décidé de conserver cette technologie pour ce projet.

En ce qui concerne la base de données, nous avons décidé d'utiliser rethinkdb, du fait de son fonctionnement de base de données en temps réel qui est le plus adapté à l'utilisation de notre base de données. Mais aussi son stockage sous forme de JSON qui le rendait facilement insérable dans nos fonctions précédemment implémentées.

3- Architecture Mobile

Utilité des différents fichiers :

- App.js :
  Gère les différentes Routes (accès aux pages de l'application).

- assets/Card.js:
  Créer la carte dans laquelle nous retrouverons les informations du services qu'elle contient.

- assets/ConnectionInput.js:
  Contient le Contenu de la page de connexion (ex: champ de texte, bouton de connexion,...).

- assets/CreateAccountInput.js:
  Contient le Contenu de la page de création de compte (ex: champ de texte, bouton d'enregistrement,...).

- assets/Disconnect.js:
  Créée le bouton qui permettra à l'utilisateur de se déconnecter du serveur.

- assets/Footer.js:
  génère un pied de page sur les différentes pages ou le composant est appelé.

- assets/Header.js:
  génère un en-tête sur les différentes pages ou le composant est appelé.

- assets/LoginButton.js:
  Crée un bouton permettant de soumettre au serveur le compte avec lequel le client tente de se connecter et le redirige ensuite vers l'application si le compte est validé.

- assets/ReactionCard.js:
  Créer une carte par service disponible sur le serveur et permet d'activer ou désactiver les réactions de celui-ci.

- assets/RegisterButton.js:
  Crée un bouton permettant de soumettre au serveur le compte que le client à créé et le redirige ensuite vers l'application si le compte est validé.

- assets/Switch.js:
  Crée un bouton qui passe d'un état activé à désactivé et vice-versa afin d'activer les actions liées au service sélectionné.

- assets/SwitchReaction.js:
  Crée un bouton qui passe d'un état activé à désactivé et vice-versa afin d'activer les réactions liées au service sélectionné.

- connection/Connection.js:
  Permet à l'utilisateur de se connecter

- connection/Register.js:
  Permet à l'utilisateur de se créer un compte.

- Oauth/Facebook.js
  Permet à l'utilisateur de lier son compte Facebook à notre application.

- Oauth/Google.js
  Permet à l'utilisateur de lier son compte Google à notre application.

- Oauth/Microsoft.js
  Permet à l'utilisateur de lier son compte Microsoft à notre application.

- appPage/Add.Js:
  Permet à l'utilisateur d'activer / désactiver des réactions du service sélectionné.

- appPage/HomePage.js:
  Permet à l'utilisateur d'activer / désactiver des réactions du service sélectionné.

- appPage/Settings.js:
  Permet à l'utilisateur de se déconnecter de l'application.

4- Architecture Web

Utilité des différents fichiers :

- App.js:
  Gère les différentes Routes (accès au page de l'application).

- Auth.js:
  Contient la page d'authentification de notre application.

- Register.js:
  Contient la page d'enregistrement de notre application.

- Home.js:
  Contient la page principale de notre application. On y trouve toutes les actions possibles sur notre application et un bouton pour s'abonner.

- Config.js:
  Contient la page de configuration de nos actions. C'est ici que l'utilisateur peut décider des réactions qui suivent les actions.

- Oauth/oauth.js:
  Contient la page de connexion au différents services que notre application supporte. L'utilisateur peut lier l'application avec les différents services proposés.

- Navbar.js:
  Contient la barre de navigation de l'application.

5- Architecture Serveur

Utilité des différents fichiers :

- index.js :
  Gère les différentes Routes/Requêtes émises par les clients, et s'occupent des actions associé (Post/Get)
- communicateDB.js :
  Gère toutes les différentes interactions avec le serveur
- apiRequest.js :
  Gère les différents appels aux APIs pour les Action/Réaction de chaques utilisateur

6- Librairies Utilisées

6.1. Mobile

- react
- react-native
- expo-google-fonts
- expo-app-loading
- expo-facebook
- expo-google-app-auth
- expo-web-browser
- expo-auth-session
- expo
- react-navigation
- Axios

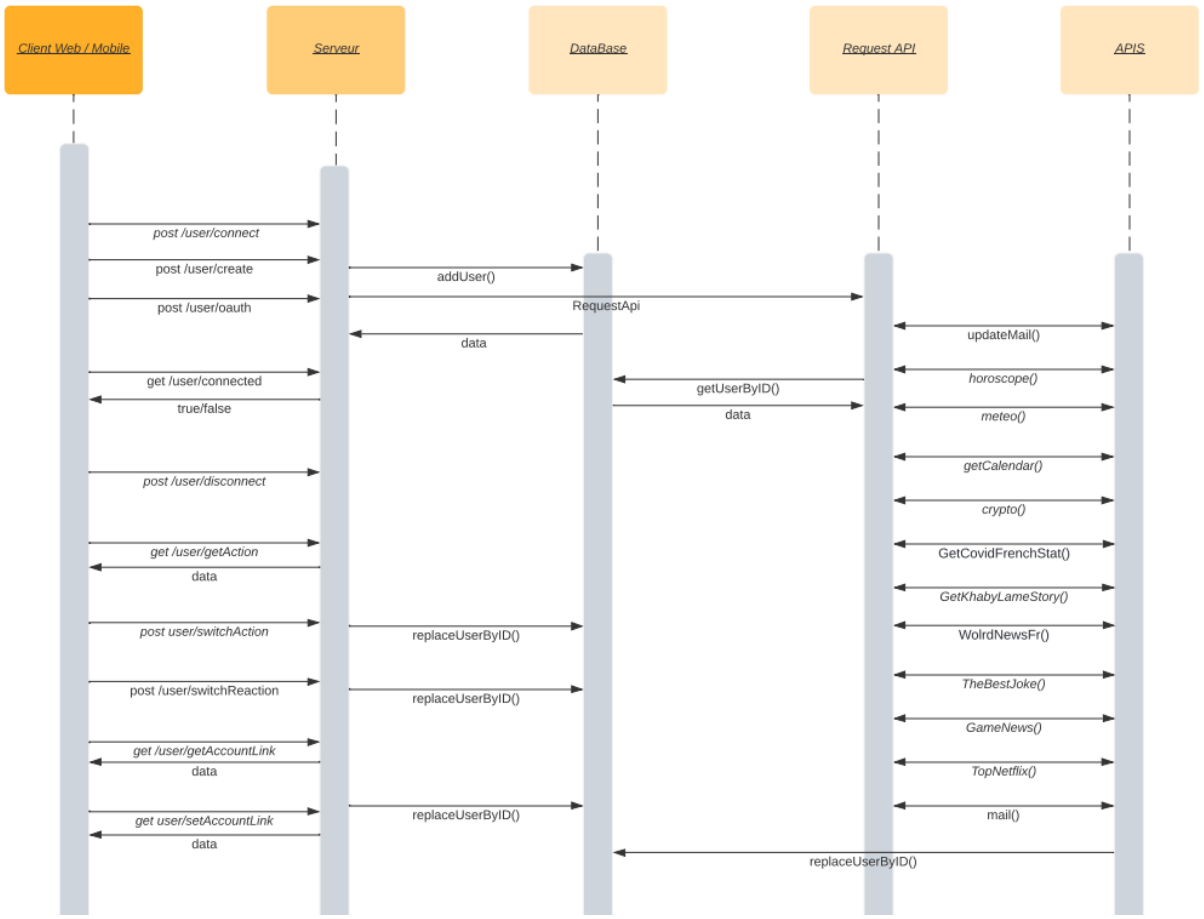
6.2. Web

- react
- react-microsoft-login
- react-facebook-login
- react-google-login
- react-router-dom
- axios

6.3. Serveur

- express
- rethinkdb
- axios

7- Diagramme de séquence



Nathan Hoche, Théo Bares,
Sevan Sarikaya & José Rodrigues