



Area

Introduction	1
Usage	2
Choix Technologique	2
Authentification via Auth0	2
Serveur en NodeJs	4
Web en Javascript	4
Mobile en React Native	4
Organisation	5
Les rôles	5
Méthodologie :	5
Les outils de collaborations :	6
Trello :	6
VsCode Share :	6
Discord :	6
Google Docs :	6
Structure du projet	6

Introduction

Area est une application serveur proposant un outil d' Actions / Réactions sur une multitude de services, accessible par authentification.

Elle dispose d'un client Web et mobile iOS / Android.

Usage

Télécharger le [Repository](#) Github:

<https://github.com/EpitechPromo2024/B-YEP-500-NCE-5-1-area-hugo.suzanne>

Assurez vous d'avoir installer nodeJs:

```
> npm install
```

Lancer l'application WEB :

```
> ./exec.sh (sur Linux / MacOS)
```

```
> ./exec.ps1 (sur Windows)
```

Puis, sur votre navigateur:

localhost:8080

Lancer l'application MOBILE :

Référez vous au README pour l'installation, puis :

```
> ./yarn run ios
```

```
> ./yarn run Android
```

Choix Technologique

- Authentification par Auth0

L'authentification est gérée grâce à l' Auth0, permettant d'autoriser l'utilisateur à accéder à l'application.

Auth0 est un service SaaS vous permettant de déléguer toute la partie authentification, gestion des rôles/permissions et du SSO de vos applications.

Leur système est très éprouvé depuis sa conception. Et c'est l'un des plus connus à ce jour.

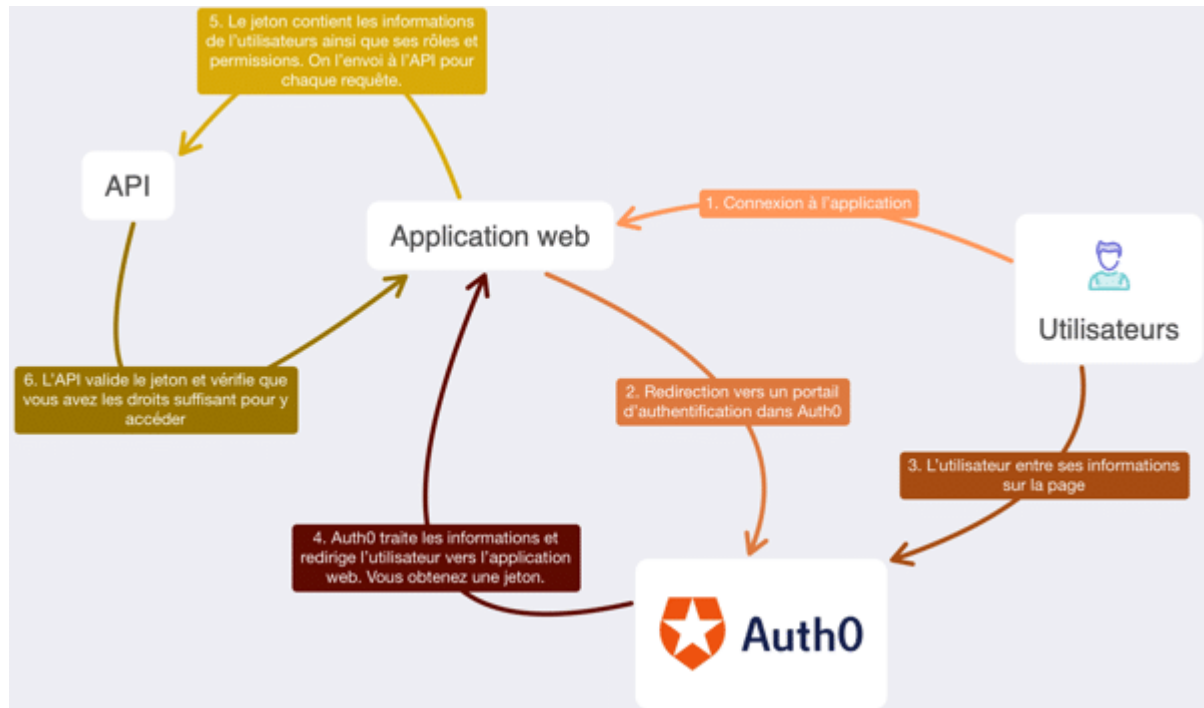
Pourquoi recourir à un système externe pour gérer son authentification ?

Pour ne pas forcément réinventer la roue continuellement, car au final tous les systèmes d'authentification restent assez semblables dans leur fonctionnement.

On renseigne un login et mot de passe et suivant les droits sur votre profil, vous pouvez accéder à certaines parties d'une app ou faire des actions.

Les avantages de passer par un service d'authentification sont assez multiples:


- rapidité d'implémentation dans votre écosystème
- gestion et implémentation centralisée
- toujours à la pointe en termes de sécurité
- permet éventuellement de faire du prototypage
- gain de temps en terme de maintenance de votre système



























L'utilisateur peut choisir de s'authentifier dans notre cas par : Google, Yammer, Github ou Facebook.

Social Connections

Configure social connections like Facebook, Twitter, Github and others so that you can let your users login with them. [Learn more](#) -

 **New!** Learn about creating [custom OAuth2 connections](#).

  TRY >	 	 
 	 	 
 	 	 
 	 	 

- Serveur en NodeJs
- Web en HTML / CSS / Javascript
- Mobile en React Natives

Organisation

Les rôles

Les rôles ont été définis selon les affinités de chacun dans le domaine. Suite à cela, nous en avons sorti 4 rôles différents :

- Développeur Front End Mobile
- Développeur Front End Web
- Développeur Back End
- Développeur devOps

Méthodologie :

Ce projet étant plus conséquent, et n'ayant pas tous la possibilité de nous retrouver au même endroit pour travailler. Il a fallu mettre en place une méthodologie s'approchant de la méthodologie Scrum Agile. Dans notre cas, environ 2 fois par semaine, nous nous réunissons pour un meeting dans le but de s'organiser le déroulement du projet. Nous pouvons partager cette méthodologie en 3 parties :

- préparation à l'avance :
 - des points qu'on souhaite aborder.
 - des tâches que nous avons effectué
 - des soucis que l'on rencontre
- pendant le meeting :
 - Avancement global du projet
 - Proposition de solution face au problème rencontrée
 - Etablissement des prochaines tâches à faire pour d'ici le prochain meeting
- après le meeting :
 - maintien de la communication au sein de l'équipe via Discord
 - pair programming via Share de VsCode
 - mis à jour des tickets Trello pour pouvoir suivre l'avancement globale

Les outils de collaborations :

- Trello :

Utilisé afin de définir des objectifs précis, définir les rôles. Et également de pouvoir commenter et suivre notre progression et celle des collaborateurs pour une meilleure vision globale du projet.

- VsCode Share :

Utilisé pendant la phase de connexion entre le backend et le frontend.

- Discord :

Utilisé notamment pour sa messagerie instantanée, pour fournir un support à n'importe quel moment ainsi que l'organisation de réunions fréquentes de l'équipe pour examiner l'avancement du projet.

- Google Docs :

Il s'agit d'un support commun pour définir les fonctionnalités ainsi que les éléments nécessaires.

Structure du projet

A la source du repo se trouve 4 plateformes :

- Client mobile : permet d'utiliser l'application sur IOS & Android
- Client Web : permet d'utiliser l'application sur le navigateur (`localhost:8080`)
- Serveur : Lie le client web & mobile
- Trigger

L'application supporte docker.

Client Mobile

Utilisation de React Native, nous avons hésité avec flutter, mais étant donné que le react native est basé sur du javascript et que le serveur est fait en nodeJs nous nous sommes naturellement tourné vers celui-ci.

Nous avons donc une application Mobile en React Native, pour pouvoir lancer l'application mobile il faut installer Android Studio qui permet d'émuler un téléphone android, et suivre les étapes du ReadMe (dans le dossier Mobile), pour lancer le projet.

Nous avons la connexion à l'application par Auth0, relié au bouton "Login" il se trouve dans le chemin Mobile/src/login/login.js.

Lorsque l'accès est vérifié, nous pouvons accéder à la page Home.

```
const onLogin = () => {
  auth0.webAuth
    .authorize({
      scope: 'openid profile email'
    })
    .then(credentials => {
      Alert.alert('AccessToken: ' + credentials.accessToken);
      setAccessToken(credentials.accessToken);
    })
    .catch(error => console.log(error));
  navigation.navigate('Home')
};
```

La page home comporte un bouton "Log out", qui permet de se déconnecter, il est constitué presque comme le bouton "Login".

Nous utilisons '.clearSession' d'Auth0 qui permet de fermer la session.

```
const onLogout = () => {
  auth0.webAuth
    .clearSession({})
    .then(success => {
      Alert.alert('Logged out!');
      setAccessToken(null);
    })
    .catch(error => {
      console.log('Log out cancelled');
    });
  navigationn.navigate('Login')
};
```

Nous avons ensuite les formulaires (qui sont fait à partir du fichier home.js dans mobile/src/home/home.js), d'actions et de réactions, sur l'onglet "action" vous avez le choix entre différents services Gmail, Github etc.

Sur le choix réactions vous avez également le choix entre différentes réactions, avec chaque choix vous avez un formulaire différent à remplir qui s'affiche comme ci-dessous du côté front.

GitHub

Repository

GMail

Expéditeur

Destinataire

Pour récupérer les informations nous faisons un fetch de l'api du serveur, qui nous permet de récupérer les informations des formulaires.

```


var jsonall = {
  "method": "post",
  headers: {
    'Authorization': `Bearer ${accesstoken}`,
    'Accept': 'application/json',
    'Content-Type': 'application/json'
  },
}
jsonall["body"] = AcReacJson
console.log("ALL", jsonall)
fetch('http://localhost:3000/workflows/add', jsonall

```

Une fois le formulaire rempli, il faut appuyer sur le bouton "Submit" qui permet l'envoi des données au back-end.

Client Web

Après authentification, le client peut utiliser les services de l'application et choisir entre différentes actions / réactions.



Welcome to AREA

Get Log to get access to more services

Your Deployed Actions/Reaction

Click on the cross for delete and stopping the action

Action1

Action2

Action3

Action2

Action4

Timer Service

Choose your Timer Action from here from here.

Actions

Le client web est organisé en plusieurs parties :

- . HTML et CSS, proposant la structure et le design de l'application.

- . Javascript, connectant les actions et réactions, gérant les authentifications, les requêtes d'API, la liaison des fichiers json et l'interaction générale avec l'application.

```
const login = async (targetUrl) => {
  try {
    console.log("Logging in", targetUrl);

    const options = {
      redirect_uri: window.location.origin
    };

    if (targetUrl) {
      options.appState = { targetUrl };
    }

    await auth0.loginWithRedirect(options);
  } catch (err) {
    console.log("Log in failed", err);
  }
};
```

```
const logout = () => {
  try {
    console.log("Logging out");
    auth0.logout({
      returnTo: window.location.origin
    });
  } catch (err) {
    console.log("Log out failed", err);
  }
};
```

- . Docker, permettant le lancement de l'application sur votre navigateur :

localhost:8080

```
docker-compose.yml
version: '2'
services:
  web:
    build: .
    command: npm run start
    volumes:
      - ../usr/app/
      - /usr/app/node_modules
    ports:
      - "8080:8080"
```

- . Fichiers json contenant les informations d'authentification et d'utilisateurs.

Pour plus d'informations, vous pouvez consulter les fichiers README.md inclus dans les dossiers WEB ou MOBILE.
