D O C U M E N T A T I O N

Commencer

L'API Area expose toute l'infrastructure de la plateforme Area via une interface de programmation normalisée. En utilisant l'API Area, vous pouvez faire à peu près tout ce que vous pouvez faire sur l'application Area, tout en utilisant le langage de programmation de votre choix. L'API Area est une API basée sur des requêtes HTTP et des réponses JSON.
Cette version de l'API, utilise OAuth 2.0. Le moyen le plus simple de commencer à utiliser l'API Area consiste à Exécuter les requêtes dans Postman. Postman est un outil gratuit qui aide les développeurs à exécuter et à déboguer les demandes d'API. Associé au serveur (ou API en question), deux clients (web/mobile) ont été développé en parallèle. Le lancement de ceux-ci ainsi que du server s'effectue à travers Docker Compose. Cette documentation présentera les démarches et instructions nécessaires au lancement, ainsi qu'à l'utilisation de l'application. Elle présentera par la même occasion l'architecture des différents composants de l'application. Les requêtes indiquées sont également fonctionnelles et accessible si vous utilisez une application tierce sans inscription d'application. L'application a pour objet d'interconnecter des services entre eux (Intra Epitech, Outlook 365, Yammer, OneDrive, etc.). Il sera donc nécessaire de demander aux utilisateurs de s'abonner à ces services. Les services disponibles à sont extraits de la liste de services implémentée côté serveur de l'application. Chaque service se voit offrir un ensemble de réaction pour chaque action effectuée sur un service connecté.

Points de terminaison

L'accès à l'API s'effectue en adressant des requêtes HTTP à une URL de nœud final de version spécifique, dans laquelle les variables GET ou POST contiennent des informations sur les éléments auxquels vous souhaitez accéder.

Réponses

Chaque réponse est encapsulée dans une balise de données. Cela signifie que si vous avez une réponse, elle sera toujours dans le champ de données. Nous incluons également un code d'état et un indicateur de réussite dans la réponse.

Lancement de l'application

L'application entière repose sur Docker Compose.

- https://docs.docker.com/compose/install/

Exécutez la commande suivant à la racine de l'application pour lancer le serveur, ainsi que les clients web/mobile.

```
➢ docker-compose up --build
```

Lancement du client Web

Rendez-vous sur l'url suivant pour accéder à l'interface web de l'application.

```
➢ http://localhost:8081
```

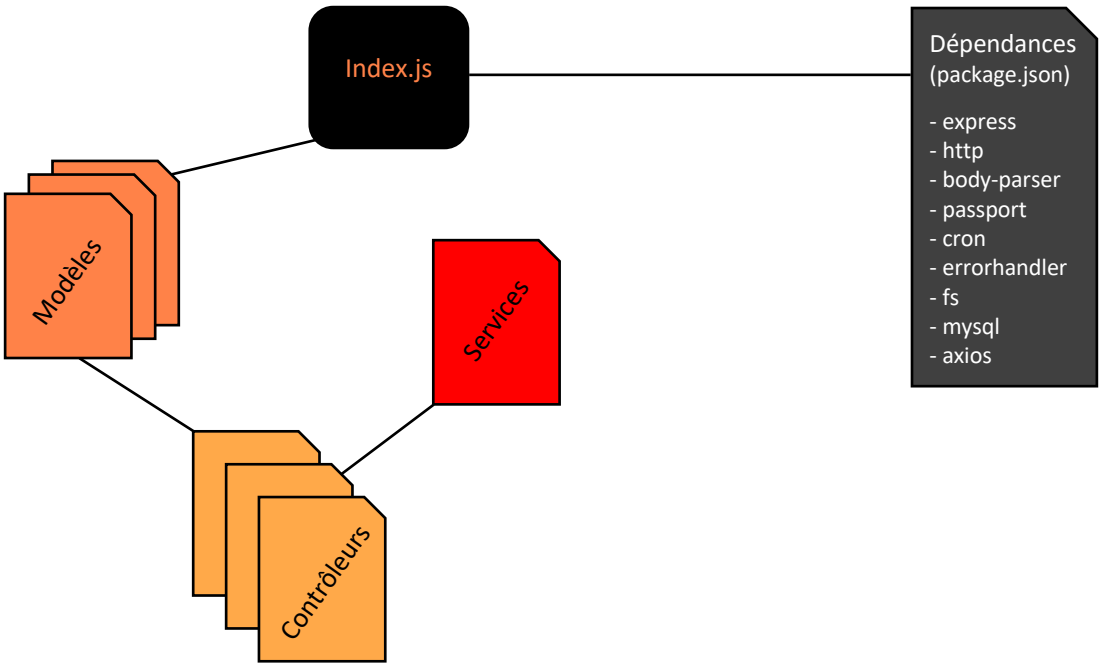
Lancement du client Mobile

Rendez-vous sur l'url suivant pour fournir la version Android du client mobile

```
➢ http://localhost:8081/client.apk
```

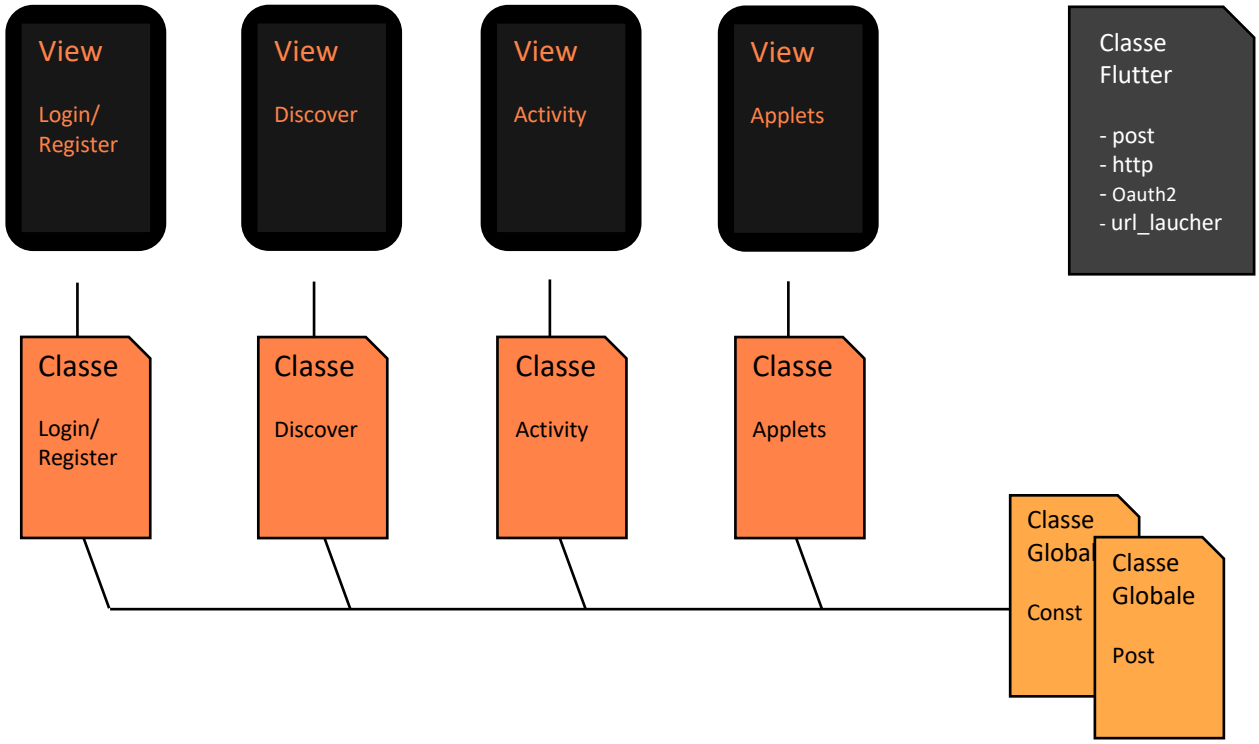
Architecture Serveur

Le serveur d'applications est le seul à intégrer la logique métier du projet. Il offre ses services aux clients Web et Mobile via une API REST.
Le serveur est accessible sur l'url suivant : http://localhost:8080/
Aucun plugin comme Sails ou architecture MVC n'a été adopté dans la structuration du serveur. Toute la logique de routage et de gestion de données a été produit en interne, voir la présentation ci-dessous.



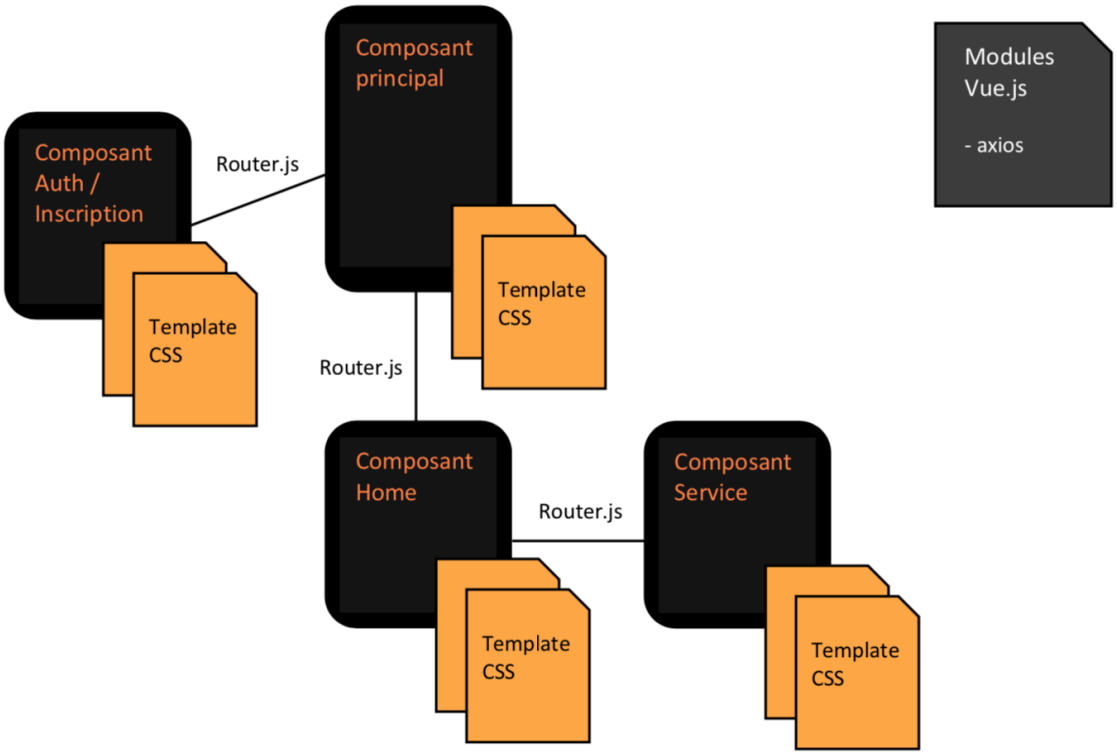
Architecture Client Mobile

Le client Mobile être disponible sur Android ou Windows Mobile. Son interface a été développé en Flutter. Aucune Template où Plugin n'est utilisé pour l'architecture, celle-ci à été développé localement. Chaque View se voit associé à une classe qui regroupe un ensemble de méthodes pour récupérer et traduire les données provenant du serveur, puis aux classes globales, accessibles depuis tous les composants de l'application.



Architecture Client Web

Le client Web est accessible sur le port 8081. Développé en Vue.js, sa structuration suit la logique de composants communiquant entre eux.



Compte

SIGN : Inscription POST

➢ http://localhost:8080/sign

Body :

Nom	Description
“firstname“	*Prénom de l’utilisateur*
“lastname“	*Nom de l’utilisateur*
“email“	*Email de l’utilisateur*
“pass“	*Mot de passe de l’utilisateur*

Response :

Status code	Message
200	“Signin successfully“

Connexion locale POST

➢ http://localhost:8080/log?method={{method}}

Url parameters :

Paramètre	Message
Method	“app“ / “google“ / “facebook“

Body :

Nom	Case	Description
“firstname“	*“google“ / “facebook“*	*Prénom de l’utilisateur*
“lastname“	*“google“ / “facebook“*	*Nom de l’utilisateur*
“email“	*“app / “google“ / “facebook“*	*Email de l’utilisateur*
“pass“	*“app / “google“ / “facebook“*	*Mot de passe de l’utilisateur*
“token“	*“google“ / “facebook“*	*Token délivré par le service*
“user_id“	*“google“ / “facebook“*	*User id délivré par le service*

Response :

Status code	Case	Message
200	“app“	“Login successfully, token generated sucessfully“
200	*“google“ / “facebook“*	“Signin sucessfully (by network)“
200	*“google“ / “facebook“*	“Login successfully, token generated sucessfully (by network)“

Connexion Facebook/Google POST

➢ http://localhost:8080/auth/facebook?queryParams={{queryParams}}

Url parameters :

Paramètre	Message
queryParams	{{User Area token}}

Body :

Nom	Case	Description
“firstname“	*“google“ / “facebook“*	*Prénom de l’utilisateur*
“lastname“	*“google“ / “facebook“*	*Nom de l’utilisateur*
“email“	*“app / “google“ / “facebook“*	*Email de l’utilisateur*
“pass“	*“app / “google“ / “facebook“*	*Mot de passe de l’utilisateur*
“token“	*“google“ / “facebook“*	*Token délivré par le service*
“user_id“	*“google“ / “facebook“*	*User id délivré par le service*

Response :

Status code	Case	Message
200	“app“	“Login successfully, token generated sucessfully “
200	*“google“ / “facebook“*	“Signin sucessfully (by network) “
200	*“google“ / “facebook“*	“Login successfully, token generated sucessfully (by network) “

Connexion services

Connexion Facebook POST

➢ http://localhost:8080/auth/facebook?queryParams={{token}}

Url parameters :

Paramètres	Contenu
queryParams	{{User Area token}}

Response :

Status code	Message
200	“Facebook token generated sucessfully”

Connexion Google POST

➢ http://localhost:8080/auth/google?queryParams={{token}}

Url parameters :

Paramètres	Contenu
queryParams	{{User Area token}}

Response :

Status code	Message
200	“Google token generated sucessfully”

Connexion GitHub POST

➢ http://localhost:8080/auth/github?queryParams={{token}}

Url parameters :

Paramètres	Contenu
queryParams	{{User Area token}}

Response :

Status code	Message
200	“GitHub token generated sucessfully”

Connexion Instagram POST

➢ http://localhost:8080/auth/instagram?queryParams={{token}}

Url parameters :

Paramètres	Contenu
queryParams	{{User Area token}}

Response :

Status code	Message
200	“Instagram token generated sucessfully”

Connexion Discord POST

➢ http://localhost:8080/auth/discord?queryParams={{token}}

Url parameters :

Paramètres	Contenu
queryParams	{{User Area token}}

Response :

Status code	Message
200	“Discord token generated sucessfully”

Connexion Slack POST

➢ http://localhost:8080/slack

Body :

Nom	Description
“token”	Token de l’utilisateur

Response :

Status code	Message
200	“Slack token generated sucessfully”

Connexion Spotify POST

➢ http://localhost:8080/spotify

Body :

Nom	Description
“token”	Token de l’utilisateur

Response :

Status code	Message
200	“Spotify token generated sucessfully”

Get service token POST

➢ http://localhost:8080/service-token

Body :

Nom	Description
“token”	Token de l’utilisateur
“userToken”	Token renvoyé par le service
“userId”	User Id renvoyé par le service
“googleMail”	Adresse mail (service Google uniquement)

Response :

Status code	Message
200	“Token added sucessfully”

Webhook (Action / Réaction)

Action Message Slack

Réaction Ajout d’une musique dans la librairie Spotify

Rédiger un message Slack commençant par “/spotify “ suivant du lien d’un morceau Spotify entraine la réaction suivante : Une musique sera ajoutée dans la librairie personnelle de l’utilisateur.

Action Notification Google

Réaction Message Slack

Lorsque l’utilisateur reçoit un mail sur Gmail, relié à son compte Google, un message est posté sur Slack.

Action Réaction Slack

Réaction Envoi de mail

Lorsqu’une réaction est ajoutée sur un message Slack, un mail est envoyé sur sa boîte mail.

Action Fichier upload sur Slack

Réaction Fichier upload sur Google Drive

Lorsqu’un fichier est upload sur Slack, il est alors upload sur Google Drive par la même occasion.

Action Dayjob

Réaction Ajout d’une playlist dans la librairie Spotify

Choisir une date dans un sélecteur sur l’application entrainera la création d’une playlist Spotify pour la même date en question.

Action Timer

Réaction Spotify

Toutes les x sec/min/j, créé une playlist Spotify.