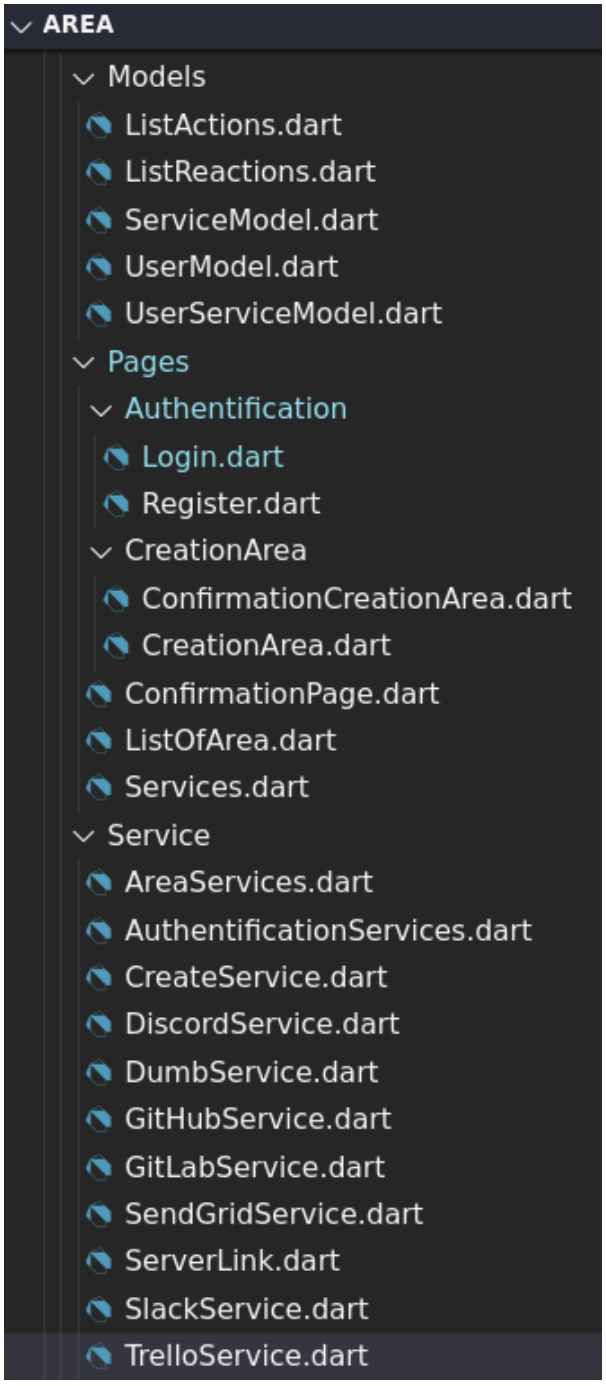
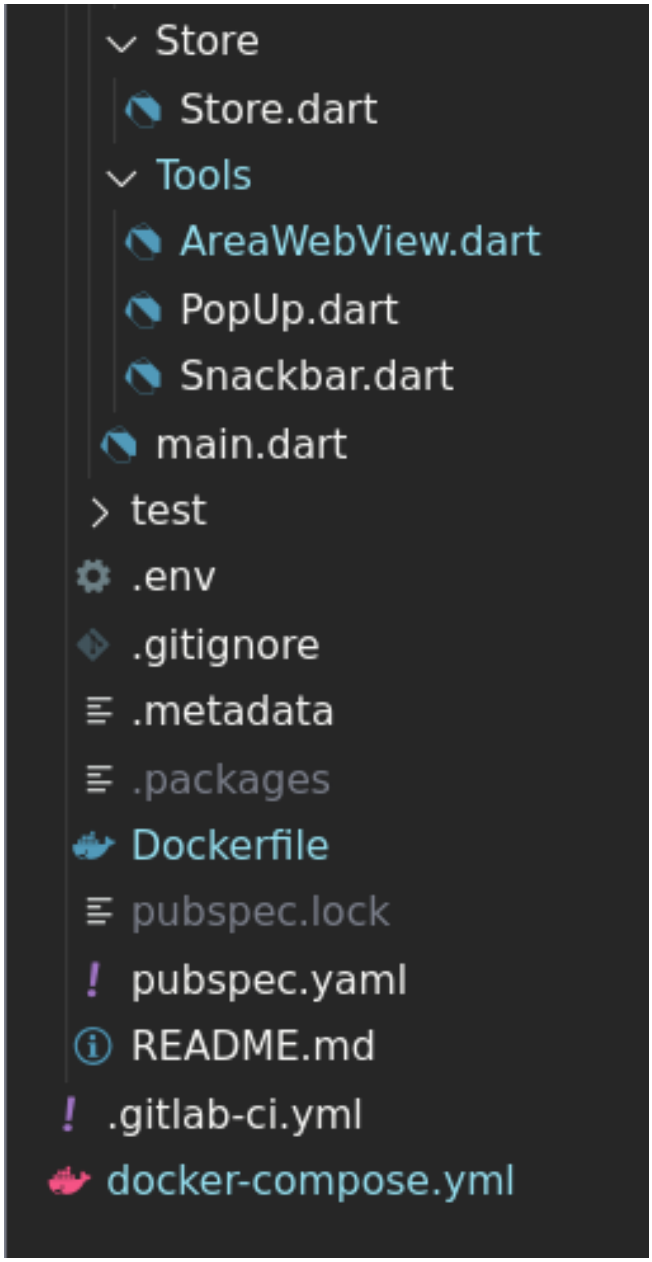
DOCUMENTATION TECHNIQUE MOBILE

Area

Coulibaly Yannis, Fontaine Melvyn,
Aflalo Gary, Corentin Calvo Wang
Zhiwen, Zhou Tony

ARCHITECTURE





ARCHITECTURE

↳ MODELS/

Ce dossier contient tous les modèles utilisés dans l'application. Les modèles sont principalement des Class.

ARCHITECTURE
↳ Models/
↳ ListActions.dart

```dart
class ListActions {
  final List<String> firstActions;
  final List<String> secondActions;

  ListActions({this.firstActions, this.secondActions});

  factory ListActions.fromJson(
      Map<String, dynamic> json, String firstService, String secondService) {
    var actionsList = ListActions(
      firstActions: json["servicesAction"][firstService][0].cast<String>(),
      secondActions: json["servicesAction"][secondService][0].cast<String>(),
    );
    return actionsList;
  }
}
```

Cette class est le modèle de la list des actions retournés par le serveur.

Il contient deux listes de String, avec dedans les noms des actions sélectionés sur les deux services.

ARCHITECTURE
↳ Models/
↳ ListReactions.dart

```dart
class ListReactions {
  final List<String> firstReactions;
  final List<String> secondReactions;

  ListReactions({this.firstReactions, this.secondReactions});

  factory ListReactions.fromJson(
      Map<String, dynamic> json, String firstService, String secondService) {
    var reactionsList = ListReactions(
      firstReactions: json["servicesReaction"][firstService][0].cast<String>(),
      secondReactions:
          json["servicesReaction"][secondService][0].cast<String>(),
    );
    return reactionsList;
  }
}       You, 15 days ago • [FEAT](mobile): add the creationArea page
```

Cette class est le modèle de la list des reactions retournées par le serveur.

Il contient deux listes de String, avec dedans les noms des réactions sélectionées sur les deux services.

ARCHITECTURE
↳ Models/
↳ ServiceModel.dart

```dart
class ServiceGroup {
  const ServiceGroup(this.item, this.group);

  final String item;
  final String group;
}
```

Cette class est le modèle des services pour savoir quelle action / reaction est lié à quel service.

- "item" correspond à l'élèment, dans notre cas une action ou une reaction.
- "group" correspond au nom du service par lequel l'action ou la réaction provient.

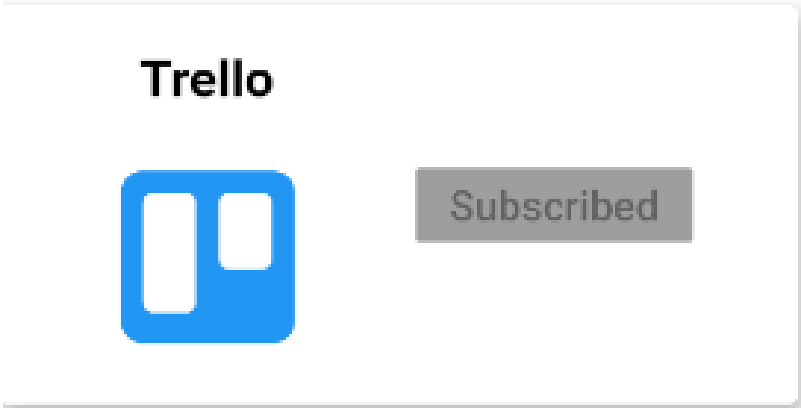
ARCHITECTURE
↳ Models/
↳ ServiceModel.dart

```dart
typedef void MyCallback();

class ServiceInfo {
  final String name;
  final IconData icon;
  final Color color;
  final bool isAvailable;
  MyCallback callback;
  ServiceInfo(
      {this.name, this.isAvailable, this.icon, this.color, this.callback});
}
```

Cette class est le modèle des services dans la page de Services, plus précisement sur la page pour s'abonner à des services.

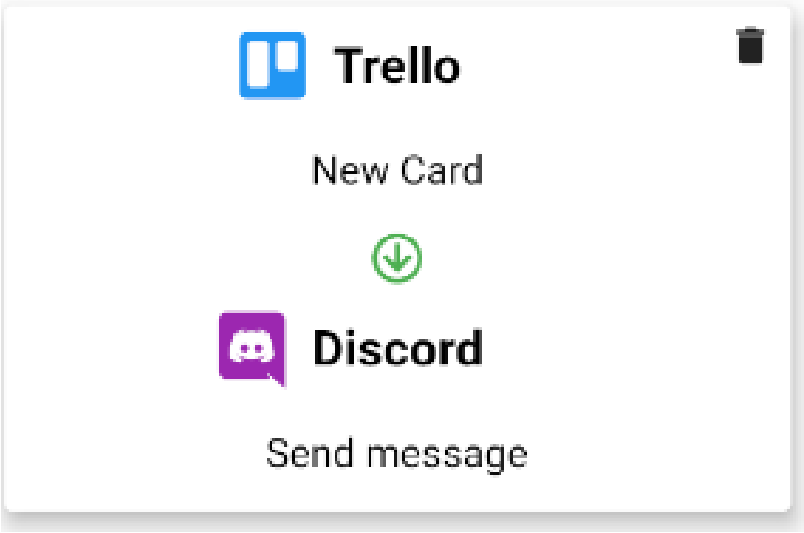
Il y a un type "MyCallback()" qui correspond à une fonction void qui ne prend aucun paramètre qui sert de callback lorsque l'utilisateur appuiera sur le bouton de souscription.



ARCHITECTURE
↳ Models/
↳ ServiceModel.dart

```dart
class AreaInfo {
  final String name;
  final IconData icon;
  final Color color;
  final String actionType;
  AreaInfo({this.name, this.icon, this.color, this.actionType});
}
```

Cette class est le modèle d'une Area crée par l'utilisateur.



ARCHITECTURE
↳ Models/
↳ UserModel.dart

```dart
class User {
  final String email;
  final String pseudo;
  final String token;

  User({this.email, this.pseudo, this.token});

  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      email: json['email'],
      pseudo: json['pseudo'],
      token: json['token'],
    );
  }
}
```

Cette class est le modèle de l'utilisateur, utilisé lorsqu'il se connecte.

ARCHITECTURE
↳ Models/
↳ UserServiceModel.dart

```dart
class UserService {
  final String id;
  final String name;
  final int nbrArea;

  UserService({this.id, this.name, this.nbrArea});

  factory UserService.fromJson(Map<String, dynamic> json) {
    return UserService(
      id: json['id'],
      name: json['name'],
      nbrArea: json['nbrArea'],
    );
  }
}      You, 5 days ago • [UPDATE](mobile): add the Confirmati

You, 5 days ago | 1 author (You)
class UserServiceGroup {
  const UserServiceGroup(this.item, this.group);

  final UserService item;
  final String group;
}
```

La class "UserService" contient les services que l'utilisateur a selectionné dans la premier page de creation d'Area.

La class "UserServiceGroup" contient "UserService" et le nom du service

ARCHITECTURE
↳ PAGES/

Ce dossier contient toutes les views où l'utilisateur peut naviguer.

ARCHITECTURE

↳ PAGES/

↳ AUTHENTIFICATION/

Ce dossier contient toutes les views où l'utilisateur peut naviguer en rapport avec l'authentification.

ARCHITECTURE

↳ Pages/

↳ Authentification/

↳ Login.dart

Cette page contient la view de la page de connexion.

ARCHITECTURE

↳ Pages/

↳ Authentification/

↳ Register.dart

Cette page contient la view de la page d'inscription.

ARCHITECTURE

↳ PAGES/

↳ CREATIONAREA/

Ce dossier contient toutes les views où l'utilisateur créer son Area, de la page ou il sélectionne ces services à la page des settings des triggers.

ARCHITECTURE

↳ Pages/

↳ CreationArea/

↳ CreationArea.dart

Cette page contient la view où l'utilisateur peut choisir quelle action et quelle reaction liées.

ARCHITECTURE

↳ Pages/

↳ CreationArea/

↳ ConfirmationCreationArea.dart

Cette page contient la view où l'utilisateur va choisir les options de chaque action et reaction. Aussi appelé trigger.

ARCHITECTURE
↳ Pages/
↳ ConfirmationPage.dart

Cette page est une page générique qui permet de créer une page de confirmation suite à une action. Il suffit lui envoyer en paramètre:
- String: message
- String: redirectPath

En effet le message sera affiché tandis que le redirectPath menera au path indiqué lorsque l'utilisateur appuira sur l'écran de confirmation.

ARCHITECTURE
↳ Pages/
↳ ListOfArea.dart

Cette page contient la view qui permet de voir la liste des Area crée par l'utilisateur. On peut aussi supprimer une area depuis cette page.

ARCHITECTURE

↳ Pages/

↳ Services.dart

Cette page la view où l'utilisateur pourra s'incrire à différent service avant de pouvoir set une Area.

ARCHITECTURE
↳ SERVICE/

Ce dossier contient tous les calls api du serveur faites par l'application.

ARCHITECTURE
↳ Service/
↳ AreaService.dart

Ce fichier contient tous les call api concerant l'area en general.

fetchAllServices(): Permet de récupérer tous les services existantes dans l'application et les services où l'utilisateur ne s'est pas encore inscrit. L'object de retour:

```
{"available": serviceNotSubscribed, "allServices": allServices};
```

fetchUserServices(): Permet de récupérer tous les services où l'utilisateur s'est abonné.

fetchUserArea(): Permet de récupérer les Area crée par l'utilisateur.

fetchDeleteArea(String areaId)**:** Permet de supprimer une Area.

fetchActions(String firstService, String secondService):
Permet de récuper les actions liés aux services et renvoie le model "**ListActions**".

fetchReactions(String firstService, String secondService):
Permet de récuper les reactions liées aux services et renvoie le model "**ListReactions**".

setServicesLink(serviceActionName, serviceActionId, actionType, , serviceReactionName, serviceReactionId, reactionType, [title]) : Permet de créer une Area.

ARCHITECTURE

↳ Service/

↳ AuthentificationService.dart

fetchLogin(String email, String password): Permet vérifier si le les credentials de l'utilisateur son corrects.

fetchRegister(String_pseudo, String_email, String_password): Permet d'envoyer l'inscription de l'utilisateur.

ARCHITECTURE
↳ Service/
↳ CreateService.dart

Ce fichier contient les requêtes "setService" des services avec un setService particulier. Les services sont:
- Twilio
- SendGrid

ARCHITECTURE
↳ Service/
↳ DiscordService.dart

Ce fichier contient les requêtes à propos du service Discord.
Il contient les call suivants:

discordSetService(data): Permet de set le service Discord.

discordGetServers(): Permet de récupérer les serveurs de l'utilisateur.

discordGetChannels(serverId): Permet de récupérer les channels d'un serveur de l'utilisateur.

ARCHITECTURE
↳ Service/
↳ DumbService.dart

Comme son nom l'indique, c'est une fichier dumb. Voyez par vous même.

```dart
dynamic dumbSetService(data) {
  return {"_id": "I'm dumb."};
}
```

C'est un HACK, pour les services qui n'ont pas besoin de setService.

ARCHITECTURE
↳ Service/
↳ GitHubService.dart

Ce fichier contient les requêtes à propos du service GitHub.
Il contient les call suivants:

githubSetService(data): Permet de set le service GitHub.

githubGetUserRepo(): Permet de récupérer les repository de l'utilisateur

ARCHITECTURE
↳ Service/
↳ GitLabService.dart

Ce fichier contient les requêtes à propos du service GitLab.
Il contient les call suivants:

gitlabSetService(data): Permet de set le service GitLab.

gitlabGetUserRepo(): Permet de récupérer les repository de l'utilisateur

ARCHITECTURE
↳ Service/
↳ SendGridService.dart

Ce fichier contient les requêtes à propos du service GitLab.
Il contient le call suivant:

sendGridSetService(data): Permet de set le service SendGrid.

ARCHITECTURE
↳ Service/
↳ ServerLink.dart

Contient une variable "serverLink" qui permet de set le lien du serveur back.

ARCHITECTURE
↳ Service/
↳ SlackService.dart

Ce fichier contient les requêtes à propos du service Slack
Il contient le call suivant:

slackGetRegisteredTeam(): Permet de récupérer les teams de l'utilisateur.

ARCHITECTURE

↳ Service/

↳ TrelloService.dart

Ce fichier contient les requêtes à propos du service Trello
Il contient les call suivants:

trelloSetService(data): Permet de set le service Trello.

trelloGetBoards(): Permet de récuperer tous les boards de l'utilisateur

trelloGetListOfBoards(String boardId): Permet de récuperer toutes les listes d'une board.

ARCHITECTURE
↳ STORE/

Ce dossier contient les fichiers en rapport avec le SharedPreferenced. On l'a renommé en Store pour que ce soit plus clair.

ARCHITECTURE

↳ Store/

↳ Store.dart

storeStringValue(String key, String value): Permet de stocker une string.

getStringValue(String key): Permet de récuperer une valeur dans le store.

removeStringValue(String key): Permet de supprimer une valeur stockée.

ARCHITECTURE
↳ TOOLS/

Ce dossiers tous les outils utile, ce sont tous des Widgets.

ARCHITECTURE
↳ Tools/
↳ AreaWebView.dart

Ce widget permet de créer une webview personnalisé.
Voici sa signature:

```dart
class AreaWebView extends StatelessWidget {
  final String title;
  final String pageUrl;
  final Completer<WebViewController> _controller =
      Completer<WebViewController>();

  AreaWebView(this.title, this.pageUrl);
```

Il prend un titre et la page URL de la webview.
A noter que cette page se ferme toute seule et ouvre une page de confirmation a la fin. Elle se ferme lorsque l'url correspond à l'url du front + /create.

Elle est utilisé pour l'OAuth2 de chaque service.

ARCHITECTURE
↳ Tools/
↳ PopUp.dart

Ce widget permet de créer une popUp personnalisé.
Voici sa signature:

```
Widget popUpCreateServicecDialog(
    BuildContext context, title, labelText, callback) {
  TextEditingController textController = TextEditingController();
```

Il prend le context, un titre, un message de description et enfin une fonction callback.

Cela affiche une popUp avec un TextField qui permet de rentrer un texte de la part de l'utilisateur. Le callback est alors appliqué en envoyant ce le texte rentré.

ARCHITECTURE

↳ Tools/

↳ Snackbar.dart

Cet outil permet de faire trigger une snackbar, c'est à dire une barre en dessous de l'écran avec un texte.

Voici les différentes signatures:

```
void showError(BuildContext context, String msg)
```

Il prend le context et un message. Il affiche une snackBar rouge indiquant une erreur.

```
void showInfo(BuildContext context, String msg)
```

Il prend le context et un message. Il affiche une snackBar de la couelur de l'application indiquant une information.

```
void showSuccess(BuildContext context, String msg)
```

Il prend le context et un message. Il affiche une snackBar de couleur verte indiquant un succès.

ARCHITECTURE

↳ main.dart

Le point d'entrée de l'application. C'est aussi ici que les routes sont définies mais aussi la bar de navigation.