Documentation Dev

Sommaire

Web

Description du programme

Langage/ SDK :

- Serveur : Node JS
- Client : React JS

Paquets utilisés pour le serveur :

```
"axios": "^0.26.0",
"body-parser": "^1.19.2",
"concurrently": "^7.0.0",
"cors": "^2.8.5",
"dotenv": "^14.2.0",
"es6-promise": "^4.2.8",
"express": "^4.17.3",
"firebase": "^9.6.4",
"firebase-admin": "^10.0.1",
"http": "^0.0.1-security",
"http-proxy-middleware": "^2.0.3",
"isomorphic-fetch": "^3.0.0",
"nodemailer": "^6.7.2",
"path": "^0.12.7",
"react-router-dom": "^6.2.1",
"react-twitter-embed": "^4.0.4",
"request": "^2.88.2",
"socket.io": "^4.4.1",
"socket.io-client": "^4.4.1",
"twit": "^2.2.11",
"twitter": "^1.7.1",
"unsplash-js": "^7.0.15",
"util": "^0.12.4"
```

Paquets utilisés pour le client :

```
"aos": "^2.3.4",
"axios": "^0.26.0",
"bootstrap": "^5.1.3",
"firebase": "^9.6.7",
"react": "^17.0.2",
"react-bootstrap": "^2.1.1",
"react-dom": "^17.0.2",
"react-icons": "^4.3.1",
"react-router": "^6.2.1",
"react-router-dom": "^6.2.1",
"react-scripts": "5.0.0"
```

Fonctions du client :

index.js :

- Fichier exécuté lors du lancement de l'application et contient toutes les routes du client.

/src :

- /components :
  - ActionsReactions.js :
    - Connecte l'utilisateur aux différentes API avec ses comptes

  - Export default : function ActionsReactions()
  - Import :
    - axios (axios)
    - IoIosClose (react-icons/io)
    - url (/global/variables.js)
    - ActionsReactions.css (/styles)
  - Functions :
    - handleDelete()
      - Supprime une action/réaction définie et envoie cette demande au serveur.
  - Variables :
  - Parameters : props

  - Api.js :
    - Connecte l'utilisateur aux différentes API avec ses comptes

  - Export default : function Api()
  - Import :
    - axios (axios)
    - url (/global/variables.js)
    - Api.css (/styles)
  - Functions :
    - handleAccessToken()
      - Récupère les access_token et les envoie au serveur
    - handleSubmit()
      - Envoie les données récupérées au serveur
  - Variables : access_token
  - Parameters : props

  - Footer.js :
    - Composant du pied de page

  - Export default : function Footer()
  - Import : Footer.css (/styles)
  - Functions :
    - componentDidMount()
      - Définit le bouton pour revenir en haut de page
  - Variables :
  - Parameters :

- Header.js :
  - Composant du haut de page pour les pages de navigations, ajoute un effet de scrolling et des animations de CSS

- Export default : function Header()
- Import :
  - AOS (AOS)
  - Navbar (/components/Navbar.js
  - Footer (/components/Footer.js)
  - Error.css (/styles)
- Functions : useEffect()
- Variables : isShrunk
- Parameters :

- Navbar.js :
  - Navbar pour le header, fixé en haut de page et ajoute un effet de scrolling qui change l'UI

- Export default : function Navbar()
- Import :
  - Link (react-router-dom)
  - Navbar.css (/styles)
- Functions :
- Variables :
- Parameters :

- Notifications.js :
  - Importe les notifications depuis la base de données du serveur et l'affiche en tant que liste.

- Export default : function Notifications()
- Import : Notifications.css
- Functions :
- Variables :
- Parameters : props

- Widget.js :
  - Composants des widgets affichés dans le tableau de bord pour importer et exporter des éléments afin de faire des requêtes avec le serveur.

- Export default : function Widget()
- Import :
  - Axios (axios)
  - url (/global/variables.js)
  - Dropdown (react-bootstrap)

        - Widget.css (/styles)
        - IoIosClose (react-icons/io)
    - Functions :
        - handleFillText(event)
            - Récupère les informations entrantes des champs de textes
        - handleSubmitInput(link, text, name)
            - Envoie les données de l'action (input) au serveur
        - handleSubmitButton(link, name)
            - Envoie les données de l'action (bouton) au serveur
        - handleSubmitReaction(link, type)
            - Envoie la réaction sélectionnée au serveur
        - handleUnsubscribe(e)
            - Désactive le widget
    - Variables : text
    - Parameters : props

- /pages :
    - About.js :
        - Affiche la page « à propos » pour présenter le projet.

    - Export default : function About()
    - Import :
        - AOS (AOS)
        - Navbar (/components/Navbar.js
        - Footer (/components/Footer.js)
        - About.css (/styles)
    - Functions : useEffect
    - Variables : isShrunk
    - Parameters :

    - Contact.js :
        - Affiche la page pour nous contacter.

    - Export default : function Contact()
    - Import :
        - AOS (AOS)
        - Navbar (/components/Navbar.js
        - Footer (/components/Footer.js)
        - Contact.css (/styles)
    - Functions : useEffect
    - Variables : isShrunk
    - Parameters :

- Dashboard.js :
  - Page principal lorsque l'utilisateur se connecte, regroupe un ensemble de composant tels que les widgets. Cette page interagit le serveur pour recevoir des données.

- Export default : function Dashboard()
- Import :
  - AOS (AOS)
  - Navbar (/components/Navbar.js
  - Footer (/components/Footer.js)
  - Dashboard.css (/styles)
- Functions :
  - handleNotifications()
    - Affiche ou non les notifications
  - handleSettings()
    - Affiche ou non les paramètres
  - handleHelp()
    - Affiche ou non les aides
  - handleAddAPI()
    - Affiche ou non les API
  - handleNotif()
    - Récupère les notifications du serveur
  - handleRefresh()
    - Récupère des infos au serveur pour actualiser les données
  - fetchDataFromAPI()
    - Récupère les infos au serveur
- Variables : showNotifications, showSettings, showHelp, showAddAPI, data, nbNotif, listnotif, state, userEmail, currentUserData
- Parameters :

- Error.js :
  - Page d'erreur lorsque la route demandée n'existe pas.

- Export default : function Error()
- Import :
  - AOS (AOS)
  - Navbar (/components/Navbar.js
  - Footer (/components/Footer.js)
  - Error.css (/styles)
- Functions : useEffect
- Variables : isShrunk
- Parameters :

- Help.js :
  - Page d'aide pour informer l'utilisateur à indiquer les étapes à suivre pour utiliser les widgets.

- Export default : function Help()
- Import : Help.css (/styles)
- Functions :
- Variables :
- Parameters :

- Home.js :
  - Page d'accueil de l'application présentant ses fonctionnalités.

- Export default : function Home()
- Import :
  - AOS (AOS)
  - Navbar (/components/Navbar.js
  - Footer (/components/Footer.js)
  - Home.css (/styles)
- Functions : useEffect
- Variables : isShrunk
- Parameters :

- Login.js :
  - Page de connexion qui communique avec le serveur.

- Export default : function Login()
- Import :
  - axios (axios
  - url (/global/variables.js)
  - useNavigate (react-router)
  - initializeApp (firebase/app)
  - getAuth (firebase/auth)
  - signInWithEmailAndPassword (firebase/auth)
  - signInWithPopup (firebase/auth)
  - GoogleAuthProvider (firebase/auth)
  - FcGoogle (react-icons/fc)
  - Button (react-bootstrap)
  - Navbar (/components/Navbar.js)
  - Login.css (/styles)
  - Link (react-router-dom)
- Functions :
  - handleEmailChange(event)
    - Récupère les informations entrantes du champ de texte pour l'email
  - handlePasswordChange(event)
    - Récupère les informations entrantes du champ de texte pour le mot de passe
  - handleSubmit(event)

        - Envoie les informations récupérées au serveur
    - handleGoogleLogin(googleE)
        - Envoie les informations au serveur pour la connexion avec Google
- Variables : email, password, error, navigate
- Parameters :

- Register.js :
    - Page d'enregistrement de comptes qui communique avec le serveur.

- Export default : function Register()
- Import :
    - getAuth (firebase/auth)
    - url (/global/variables.js)
    - createUserWithEmailAndPassword (firebase/auth)
    - sendEmailVerification (firebase/auth)
    - Button (react-bootstrap)
    - Link (react-router-dom)
    - Navbar (/components/Navbar.js)
    - Register.css (/styles)
    - Axios (axios)
- Functions :
    - handleEmailChange(event)
        - Récupère les informations entrantes du champ de texte pour l'email
    - handlePasswordChange(event)
        - Récupère les informations entrantes du champ de texte pour le mot de passe
    - handleConfirmPasswordChange(event)
        - Récupère les informations entrantes du champ de texte pour confirmer le mot de passe
    - handleSubmit(event)
        - Envoie les informations récupérées au serveur
- Variables : email, password, confirmPassword, error
- Parameters :

- Services.js :
    - Page de présentation des services proposés.

- Export default : function Services()
- Import :
    - AOS (AOS)
    - Navbar (/components/Navbar.js)
    - Footer (/components/Footer.js)
    - Services.css (/styles)
- Functions : useEffect
- Variables : isShrunk
- Parameters :

- Settings.js :
  - Page des paramètres présentant les paramètres utilisateurs définies par défaut et modifier par celui-ci.

- `Export default : function Help()`
- `Import :`
  - `Axios (axios)`
  - `Settings.css (/styles)`
- `Functions :`
  - `handleIp(event)`
    - `Récupère les informations entrantes du champ de texte pour définir l'IP`
  - `handleEmail(event)`
    - `Récupère les informations entrantes du champ de texte pour définir l'email`
  - `handlePhone(event)`
    - `Récupère les informations entrantes du champ de texte pour définir le numéro de téléphone`
  - `handleSubmit(event)`
    - `Envoie les informations au serveur`
- `Variables : ip, email, phone`
- `Parameters :`

- /styles :
  - Fichiers .css :
    - Modifie la mise en page pour chaque composant définis en fonction du nom du fichier. Ajoute des effets d'animations de dégradés de couleur, des changements de positions et de visuels.
    - Pour les animations, @keyframe permet de créer des fonctions d'animations dans le CSS. Les animations sont définies en fonction d'étapes (0%, 50%, 100% par exemple) qui peuvent bouger la position du fond dans le cas du fond animé.

Fonctions du serveur :

index.js :

- Fichier exécuté lors du lancement de l'application et contient toutes les routes du serveur accessibles par le client et initialise la base de données.
- Port : 8080

config.js :

- Fichier de configuration de Firebase à remplir via un fichier .env :
  - PORT
  - HOST
  - HOST_URL
  - API_KEY
  - AUTH_DOMAIN
  - DATABASE_URL

- PROJECT_ID
- STORAGE_BUCKET
- MESSAGING_SENDER_ID
- APP_ID
- MSM_ID

/controllers :

- animeController.js :
  - Functions :
    - setReactionUpdateAnime()
      - Récupère les données de l'API et renvoie une réaction en fonction d'une route
    - getUpdateAnime()
      - Récupère les entrées utilisateurs
    - setReactionWatchingList()
      - Récupère les données de l'API et renvoie une réaction en fonction d'une route
    - getWatchingList()
      - Récupère les entrées utilisateurs
    - setReactionPTWIsAiring()
      - Récupère les données l'API lorsqu' un anime OpenToWatch sort et renvoie une réaction
    - setReactionUpdateManga()
      - Récupère les données de l'API et renvoie une réaction en fonction d'une route
    - getUpdateManga()
      - Récupère les entrées utilisateurs
    - setReactionSeasonTop()
      - Récupère les données de l'API et renvoie une réaction
    - getSeasonTop()
      - Récupère l'entrée utilisateurs

- apiController.js :
  - getApi()
    - Récupère les clés API stockées dans Firebase

- covidController.js :
  - setReactionCovidNumber()
    - Récupère les données de l'API et renvoie une réaction
  - getNumber()
    - Procède l'automatisation de comparaison des valeurs récupérées

- cryptoController.js :
  - setReactionCrypto()
    - Récupère les données de l'API et renvoie une réaction

- getCrypto()
  - Procède l'automatisation de comparaison des valeurs récupérées

- emailController.js :
  - `emailController()`
    - Initialise la réaction d'envoie d'email et récupère les données fournies par les API
  - setEmailController()
    - Procède l'automatisation de comparaison des valeurs récupérées
  - `setEmail()`
    - Récupère l'email et défini la valeur global email
  - `getEmail()`
    - `Envoie au format JSON l'email en réponse`

- esportController.js :
  - `setReactionDayMatches()`
    - Récupère les données de l'API
  - getDayMatches()
    - Procède l'automatisation de comparaison des valeurs récupérées
  - `setReactionPastMatches()`
    - Récupère les données de l'API et renvoie une réaction
  - getPastMatches()
    - Procède l'automatisation de comparaison des valeurs récupérées
  - `setReactionTournaments()`
    - Récupère les données de l'API et renvoie une réaction
  - getTournament()
    - Procède l'automatisation de comparaison des valeurs récupérées

- footballController.js :
  - setReactionStandings()
    - Récupère les données de l'API
  - getStandings ()
    - Procède l'automatisation de comparaison des valeurs récupérées
  - setReactionScorers()
    - Récupère les données de l'API et renvoie une réaction
  - getScorers()
    - Procède l'automatisation de comparaison des valeurs récupérées
  - setReactionPastL1Matches()
    - Récupère les données de l'API et renvoie une réaction
  - getReactionPastL1Matches()
    - Procède l'automatisation de comparaison des valeurs récupérées
  - setReactionL1Matches()
    - Récupère les données de l'API
  - getL1Matches()

        - Procède l'automatisation de comparaison des valeurs récupérées
    - setReactionFootMatches()
        - Récupère les données de l'API et renvoie une réaction
    - getFootMatches()
        - Procède l'automatisation de comparaison des valeurs récupérées
    - setReactionTeamMatches()
        - Récupère les données de l'API et renvoie une réaction
    - getReactionTeamMatches()
        - Procède l'automatisation de comparaison des valeurs récupérées

- githubController.js :
    - `setReactionGithub()`
        - Récupère les données de l'API et renvoie une réaction
    - getUserGithub()
        - Procède l'automatisation de comparaison des valeurs récupérées
    - `setNameFolder()`
        - Récupère les données de l'API et renvoie une réaction en définissant un nom de dossier
    - getNameFolder()
        - Procède l'automatisation de comparaison des valeurs récupérées
    - `setAccessToken()`
        - Récupère la clé API du client

- reactionController.js :
  - `reactionController()`
    - Récupère les données de l'API et renvoie une réaction
  - `setInterval()`
    - Défini la valeur des intervalles en secondes

- twitterController.js :
  - `setReactionTwitterTrends()`
    - Récupère les données de l'API et renvoie une réaction
  - `getTwitterTrends()`
    - Procède l'automatisation de comparaison des valeurs récupérées
  - `setReactionTwitterLocation()`
    - Récupère les données de l'API et renvoie une réaction
  - `getTwitterLocation()`
    - Procède l'automatisation de comparaison des valeurs récupérées
  - `getLastTweet()`
    - Récupère les données de l'API et renvoie une réaction
  - `getTimerIntervalTwitter()`
    - Récupère la valeur définie par le client pour l'intervalle
  - `intervalTweet()`
    - Procède l'automatisation de comparaison des valeurs récupérées en fonction de la valeur de l'intervalle propre à Twitter

- unsplashController.js :
  - `setReactionUnsplash()`
    - Récupère les données de l'API et renvoie une réaction
  - `getUnsplashPicture()`
    - Procède l'automatisation de comparaison des valeurs récupérées
  - `setRandomUnsplashPicture()`
    - Récupère les données de l'API et renvoie une réaction
  - `getRandomUnsplashPicture()`
    - Procède l'automatisation de comparaison des valeurs récupérées
  - `setTimerUnsplash()`
    - Défini la valeur pour chaque intervalle d'automatisation pour Unsplash

- userController.js :
  - `addUser()`
    - Récupère les données reçues pour les ajouter dans la base de données
  - `getAllUsers()`
    - Renvoie toutes les données de tous les utilisateurs
  - `getUser()`
    - Renvoie toutes les données d'un utilisateur
  - `updateUser()`

        - Récupère les données reçues et met à jour l'utilisateur
    - `deleteUser()`
        - Supprime les données de l'utilisateur

- weatherController.js :
    - `setReactionTemp()`
        - Récupère les données de l'API et renvoie une réaction
    - `getTemp()`
        - Procède l'automatisation de comparaison des valeurs récupérées
    - `setReactionWeather()`
        - Récupère les données de l'API et renvoie une réaction
    - `getWeather()`
        - Procède l'automatisation de comparaison des valeurs récupérées
- weatherController.js :
    - `setReactionNewsbyCountry()`
        - Récupère les données de l'API et renvoie une réaction
    - `getNewsbyCountry()`
        - Procède l'automatisation de comparaison des valeurs récupérées
    - `setReactionNewsbySubject()`
        - Récupère les données de l'API et renvoie une réaction
    - `getNewsbySubject()`
        - Procède l'automatisation de comparaison des valeurs récupérées
    - `setReactionNewsbySource)`
        - Récupère les données de l'API et renvoie une réaction
    - `getNewsbySource()`
        - Procède l'automatisation de comparaison des valeurs récupérées

/db :

- area.json :
    - type
    - project_id
    - private_key_id
    - private_key
    - client_email
    - client_id
    - auth_uri
    - token_uri
    - auth_provider_x509_cert_url
    - client_x509_cert_url
- firebase.js :
    - Récupère les données du fichier config et initialise la base de données Firebase

/global :

- variables.js :
    - Stocke toutes les variables globales du serveur

/models :

- user.js :
  - Définit la structure des données pour chaque nouvel utilisateur

/routes :

- Répertorie toutes les routes en fonction des controllers. Chaques controllers possèdent leur

Lancement des programmes

Le client et le serveur se lancent avec les mêmes commandes depuis deux répertoires différents. Ils utilisent le gestionnaire de paquet npm adapté pour les applications web.

Le client :

Aller dans le répertoire /web/client. Depuis ce répertoire, exécuter la commande suivante :

« npm install && npm start »

Le serveur :

Aller dans le répertoire /web/server. Depuis ce répertoire, exécuter la commande suivante :

« npm install && npm start »

Application

Description du programme

Langage/ SDK :

- Flutter (dart)
- Version SDK minimum : 31

Paquets utilisés :

```
cupertino_icons: ^1.0.2
google_sign_in: ^5.2.3
firebase_auth: ^1.0.1
font_awesome_flutter: ^9.0.0
provider: ^5.0.0
dio: ^4.0.4
image_picker: ^0.8.4+4
```

Fonctions :

main.dart :

- Fichier exécuté lors du lancement de l'application.

app.dart :

- Stocke toutes les routes disponibles. Gère le thème de l'application et le statut de connexion de l'utilisateur.

/api :

- auth_email.dart :
  - Gère l'échange de données entre l'application et Firebase pour l'authentification par email/mot de passe.
- auth_google.dart :
  - Gère l'échange de données entre l'application et Firebase pour l'authentification par un compte Google.

/components :

- drawer.dart :
  - Composant pour afficher le volet déroulant comportant plusieurs actions disponibles.
- widget.dart :
  - Composant pour afficher un widget récupéré dans le fichier json du serveur. Appelé dans 'dashboard.dart' pour afficher plusieurs widget d'affilés.

/global :

- variables.dart :
  - Stocke des variables utiles dans plusieurs fichiers pour pouvoir les appeler plus facilement via l'import.

/pages :

- dashboard.dart :
    - Affiche l'ensemble des widgets du compte.
- home.dart :
    - Dépendant du statut de connexion de l'utilisateur, affiche soit le dashboard soit la page de connexion en fonction de celui-ci.
- login.dart :
    - Affiche un formulaire à remplir pour s'authentifier avec différents comptes.
- register.dart :
    - Affiche un formulaire à remplir pour s'inscrire par mail/mot de passe.
- settings.dart :
    - Affiche l'ensemble des paramètres utilisateurs.

/theme :

- theme.dart :
    - Fichier de configuration du thème de l'application.

Lancement du programme :

Pour lancer le programme, vous devez d'abord lancer un émulateur Android avec, de préférences, une version d'APK supérieur à Q (Android 10.0.0).

En mode debug :

« flutter pub get && flutter run »

En mode release :

« flutter pub get && flutter run --release »

L'application n'est pas destinée à être exécuter dans un navigateur !

Docker

Utiliser Docker

Description :

Dans un premier temps, Docker va générer les images des applications web (client et serveur) puis générer l'image du fichier APK (fichier de configuration des applications mobiles) de l'application mobile.

Commandes :

A la racine de dossier Area :

« docker-compose build && docker-compose up »