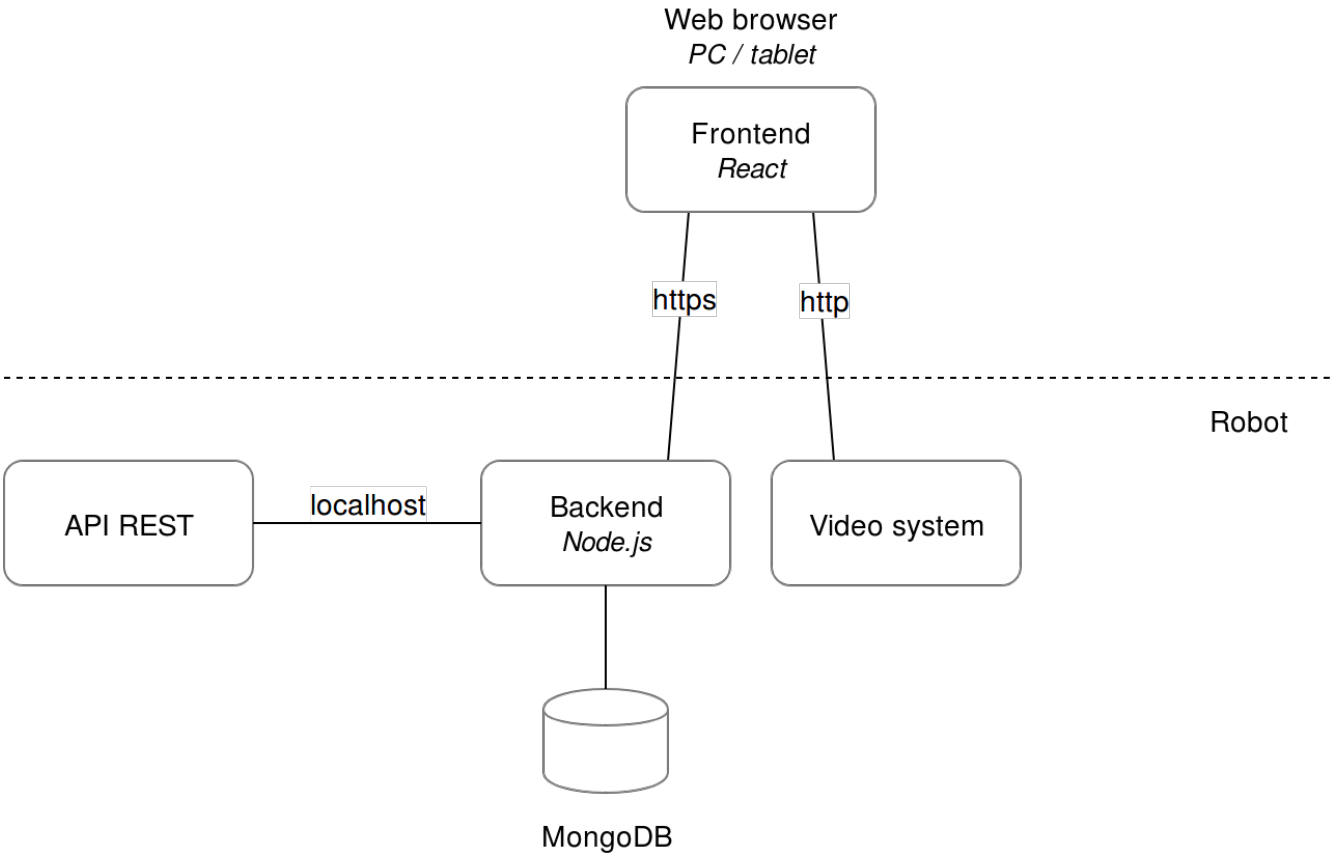
awabot

INTELLIGENCE

Econobots - Architecture

Architecture générale

La solution fournie par Awabot est une application web permettant de piloter le robot. Son principal avantage est qu'elle est multi plate-forme, c'est à dire qu'il est possible d'avoir la main sur le robot depuis n'importe quel ordinateur ou encore une tablette. Voici les composants principaux entrant en jeu dans cette application :



La brique centrale est le *Backend*, écrit en Javascript et tournant dans l'environnement *Node.js*. Sa fonction première est de servir le *Frontend* de l'application, utilisant quant à lui *React*, une technologie Javascript orientée *Component*, utilisable dans un navigateur web et très optimisée pour les changements de vue dynamiques et fréquents. Ce *Frontend* constitue l'élément principal avec lequel les utilisateurs interagiront.

Le *Backend* expose aussi une API REST permettant au *Frontend* d'utiliser toute la couche métier de l'application, comme la gestion des utilisateurs par exemple, pour laquelle il utilise une base de données *MongoDB*. Il joue également le rôle de proxy vis-à-vis de l'API REST du robot, afin de pouvoir ordonnancer et contrôler les requêtes provenant du *Frontend* de l'application.

Enfin, le *Frontend* de l'application interagit directement avec le système vidéo du robot (pour le visionage uniquement, pas pour l'enregistrement), en utilisant son API directement exposée sur HTTP, ce qui permet d'utiliser des éléments HTML standards pour ce qui est de l'affichage des flux caméra du robot.

Architecture détaillée

Backend

Commençons par nous intéresser au *Backend* de l'application. Comme mentionné plus haut, il s'agit d'un projet *Node.js*, construit principalement avec le paquet *Express* qui est l'un des plus populaires en

matière de création de serveur web en Javascript.

À la racine du dossier *backend* on trouve deux fichiers importants:

- *config.js* : contenant tous les paramètres nécessaires au fonctionnement du backend
- *package.json* : contenant la description du projet *Node.js*, notamment ses dépendances

Dossier "database"

Ce dossier contient un unique fichier Javascript *init.js*. Il permet de (ré)initialiser la base de données utilisée par le *Backend* et d'y insérer l'utilisateur administrateur principal, dont les informations sont renseignées dans le fichier *config.js*.

Dossier "ssl"

Ce dossier contient un simple script de génération de certificat SSL auto-signé qui peut être utilisé par le *Backend* afin d'activer le support d'HTTPS. Le seul inconvénient d'un tel certificat est que les navigateurs bloquent, par défaut, la connection lorqu'ils n'arrivent pas à vérifier le certificat fourni par le serveur. Il faut alors ajouter une exception pour qu'il soit possible d'accéder à l'application.

Dossier "views"

Ce dossier contient la page web principale servie par le *Backend*. Elle est extrêmement simple dû à l'utilisation de la technologie *React* : tout le *Frontend* n'est autre qu'un ensemble de scripts Javascript permettant la création de l'interface utilisateur. En mode production, il s'agit d'un seul fichier Javascript minifié, appelé en général *main.<hash>.js*, contenant toute la logique du *Frontent* de l'application, c'est à dire l'ensemble de l'interface utilisateur ainsi que tous les appels au *Backend* via *Ajax*.

Dossier "src"

On en arrive au dossier principal : le dossier contenant les sources, dont le point d'entrée est le fichier *main.js*. Ce dernier initialise tous les éléments principaux nécessaires au fonctionnement du *Backend* :

- connexion à la base de données
- définition des URL accessibles via HTTP (routes)
- démarrage du serveur HTTP (ou HTTPS suivant la configuration)

Dossier "routes"

Ce dossier contient tous les points d'entrée du *Backend* qui sont organisés en différent "routers" :

- *RootRouter.js* : servant la page web unique de l'application (présente dans le dossier *views*)
- *AuthenticationRouter.js* : exposant les URLs permettant de gérer l'authentification des utilisateurs
- *UserRouter.js* : exposant les URLs permettant de gérer les utilisateurs de la plate-forme
- *RobotRouter.js* : exposant les URLs permettant d'interagir indirectement avec le robot
- *ErrorRouter.js* : interceptant toutes les erreurs telles que *HTTP 404*, ou encore *HTTP 500*

Dossier "core"

Ce dossier contient toutes les briques servant au cœur du fonctionnement du *Backend*, nous allons parler uniquement de celles qui ont un rôle majeur :

- *DatabaseManager.js* : brique ayant la main sur la base de données et permettant au reste du *Backend* de faire des requêtes
- *PilotManager.js* : brique gérant tous les aspects relatifs au pilotage du robot, notamment le fait qu'il y ait un unique pilote ainsi que la gestion de l'enregistrement vidéo
- *RobotLink.js* : brique permettant de faire appel à l'API REST du robot
- *SessionManager.js* : brique gérant les sessions des utilisateurs connectés à la plate-forme
- *URLAccessManager.js* : brique contrôlant l'accès aux différentes URL exposées par le *Backend*, en fonction de l'utilisateur qui fait la requête

Dossier "socketio"

Ce dossier contient un unique fichier *SocketIOServer.js* contenant toute la logique de l'aspect "server push" du *Backend*. Sa principale fonction est de transmettre les diverses informations d'état du robot aux utilisateurs connectés sur l'application. Pour cela, il implémente un système de polling sur l'API REST du robot via des appels réguliers sur */state*.

Ce système n'est déclenché que lorsqu'il y a au moins un utilisateur connecté sur la page d'accueil (Si tous les utilisateurs connectés se trouvent sur la page *Account* ou *Admin*, le polling n'est pas actif pour des questions d'optimisation). La brique ne transmet aussi les informations d'état que lorsque celles-ci changent, ou que le *Frontent* les demande explicitement, ce qui permet d'éviter le transport d'informations lorsque cela est inutile.

L'autre fonction de cette brique est de constituer le point d'entrée de tout l'aspect pilotage du robot ainsi que les fonctionnalités rattachées telles que l'enregistrement vidéo par exemple.

Frontend

Tout comme le *Backend*, le *Frontent* est un projet *Node.js*, mais qui n'utilise pas les mêmes paquets. On retrouve donc le fichier *package.json* ainsi qu'un dossier de sources.

Dossier "src"

Deux fichiers sont présents à la racine du dossier de sources :

- *index.js* : point d'entrée du *Frontend*
- *App.js* : composant *React* racine, utilisé dans *index.js*, instanciant en cascade tous les autres composants de l'application

Dossier "assets"

Ce dossier contient toutes les ressources graphiques utilisées par le *Frontend* de l'application.

Dossier "i18n"

Ce dossier contient la brique chargée du mécanisme d'internationalisation du *Frontend* de l'application. Le cœur du mécanisme se trouve dans le fichier *i18n.js*, lui même chargeant les différents fichiers de langue disponibles. À l'heure actuelle, seulement le fichier *en.js* a été créé. Il s'agit d'un simple fichier ayant une architecture clé/valeur.

Pour ajouter une nouvelle langue :

- Copier le fichier *en.js* et éditer toutes les valeurs présentes
- Ajouter cette nouvelle langue dans le fichier *i18n.js*

- Implémenter la fonction *getLocale()* du fichier *i18n.js* afin que celle-ci retourne le bon code de langue, en fonction de l'utilisateur connecté par exemple

Dossier "css"

Ce dossier contient toutes les feuilles de styles utilisées par les différents composants du *Frontend* de l'application.

Dossier "common"

Ce dossier contient tous les éléments qui ne sont pas des composants *React*. On trouve les éléments principaux suivants :

- *ajax.js* : permettant de faire des requêtes HTTP asynchrones au *Backend*, à tout moment lors du cycle de vie du *Frontend* de l'application
- *config.js* : contenant un certain nombre de paramètres, notamment les résolutions des flux vidéo
- *GamepadManager.js* : brique permettant d'utiliser l'API Gamepad présente sur les navigateurs
- *SocketIOClient.js* : point d'entrée/sortie de l'aspect "server push", côté *Frontend*
- *StateManager.js* : brique réalisant un système de publisher/subscriber pour tout type d'événements

Dossier "components"

Ce dossier contient tous les composants *React* servant à l'interface utilisateur de l'application. Comme mentionné plus haut, l'instanciation des composants *React* se fait en cascade. On trouve par exemple les composants *Header.js* et *Body.js* qui sont instanciés par le composant racine *App.js*. Le composant *Body.js* quant à lui instancie des composants différents selon la page sur laquelle se trouve l'utilisateur. Il peut donc instancier un composant parmi ceux-ci :

- *Login.js*
- *Home.js*
- *Account.js*
- *Admin.js*

Et ainsi de suite jusqu'à arriver aux composants terminaux, comme par exemple *VideoPlayer.js*, qui permet d'afficher un popup de lecture vidéo, ou encore *Joystick.js* qui permet d'afficher le joystick permettant de piloter de robot.