VIDÉO EN CONTINUE

AVLAH Love Josué

2188284

VINCENT-CHARLAND Maxime

2158920

Travail présenté à

BEN-KHEDHER Dhafer

OBJETS CONNECTÉS

Groupe 1

Collège de Bois-de-Boulogne

Informatique

Jeudi 4 avril 2024

Architecture du projet

Dans un premier temps, nous avons divisé notre application en deux portions : une partie « client » et une partie « serveur. »

## Partie Client

Grosso modo, la partie client contient les différentes entités de l’application.

D’une part les entités **graphiques** en l’occurrence :

* le contrôleur de vidéos;
* le lecteur de vidéos;
* l’affiche de la date et de l’heure.

D’autre part les entités **électroniques** telles que :

* le détecteur de mouvements;
* l’émetteur de signaux lumineux.

## Partie Serveur

La partie serveur sert principalement à faire abstraction des requêtes SQL via une API REST. Le back-end utilise l’ORM Peewee.

Fonctionnalités

Voici les fonctionnalités que renferme notre application :

1. Au démarrage, une fenêtre en plein écran affichant la date et l’heure apparaît lorsqu’il n’y a pas de vidéos disponibles dans le dossier servant à les contenir.
2. En cas de présence de vidéos dans ledit dossier, un lecteur de vidéo s’affichant en plein écran les lit en continu. S’Il n’y a qu’une seule vidéo disponible, elle sera lue en boucle.
3. Les boutons du contrôleur de vidéo permettent d’effectuer les fonctions suivantes :
   * Bouton « Localisation/arrêt » : lorsque cliqué, quand une vidéo est en cours de lecture, un voyant lumineux clignote. Lorsque le bouton est cliqué de nouveau, pendant la lecture d’une vidéo, le voyant clignotement s’arrête.
   * Bouton « Passer au suivant » : permet d’aller à la vidéo suivante.
   * Bouton « Arrêter les vidéos » : permet d’interrompre la lecture de vidéo et de fermer la fenêtre du lecteur quand une vidéo est en cours de lecture.
   * Bouton « Démarrer les vidéos » : permet de lire les vidéos quand il n’y a pas de vidéo en cours de lecture.
4. Pendant la lecture d’une vidéo, si un mouvement est détecté par le capteur de mouvement, le lecteur joue la vidéo suivante.
5. Un lecteur de vidéos s’affiche en plein écran et enregistre dans une base de données les informations relatives à la lecture des vidéos en arrière-plan.
6. Le contrôleur de vidéos, affiche les informations suivantes :
   * Nom de la vidéo courante;
   * Nombre de lecture de la vidéo courante dans la journée;
   * Nombre total de vidéos jouées dans la journée;
   * Oui ou Non si le capteur de mouvement en détecte un.

# Configuration matérielle

Pour la configuration matérielle du Raspberry Pi et des capteurs GPIO, suivez les étapes d’installation du capteur IR du document Chapitre3-Tutorial-Matriel.pdf, chapitre 23.

Installation

1. Cloner le dépot GitHub

git clone [git@github.com](mailto:git@github.com):mxmvncnt/TP1-objets-connectes.git

cd TP1-objets-connectes

1. Créer le fichier .env dans le dossier ./api avec les valeurs suivantes (modifier par les valeurs appropriées):

DB\_HOST=localhost

DB\_PORT=3306

DB\_USER=utilisateur-mysql

DB\_PASS=mot-de-passe

DB\_NAME=nom-de-la-db

1. Installer les dépendances du serveur d’API et lancer le serveur

cd api

pip install flask pymysql peewee python-dotenv

flask run

1. Installer les dépendances du client

Ouvrir un autre terminal et tapez:

cd client

pip install RPi.GPIO python-vlc python-dotenv

1. Créer un fichier .env qui contient ce qui suit:

API\_URL=http://localhost:5000

1. créer un dossier vidéos dans le dossier client/

mkdir videos

1. Ajouter des vidéos au dossier videos/  
   Si vous n’ajoutez pas de vidéos, l’horloge s’affichera quand vous lancerez l’application.
2. Lancer l’application  
   python main.py

# Serveur backend

Le serveur backend est conçu avec Java et Spring Boot. Il peut donc être exécuté sur un très grand nombre de machines avec de bonnes performances grâce à la JVM. Pour exécuter le serveur, vous devez donc avoir Java ainsi que Maven d’installé sur votre machine. Vous devez aussi avoir préalablement créé une base de données avec le fichier de création situé dans tp1/src/main/resources.

# Installation

1. Cloner le dépot GitHub  
   git clone [git@github.com](mailto:git@github.com):mxmvncnt/TP1-objets-connectes.git  
   cd TP1-objets-connectes/server
2. Créer un fichier secrets.properties localisé dans tp1/src/main/resources. Pour créer ce fichier, vous pouvez aussi copier le fichier secrets.properties.template et retirer le « .template ». Dans ce fichier, ajoutez/remplacez les informations de connexion à votre base de données.
3. Installer les dépendances  
   mvn clean install
4. Exécuter le serveur  
   mvn spring-boot:run

# Interface Web

L’interface web a été développé avec le Framework Angular avec la librairie Ant Design, une boite à outils qui permet d’intégrer des composants graphiques de très haute qualité. Pour démarrer l’interface web, il faut préalablement avoir installé Node Package Manager (npm). Il faut également s’assurer d’initialiser la valeur de la variable « apiUrl » situé dans les deux fichiers du dossier web/src/environments comme suit : 'http://localhost:{numéro de port}', en spécifiant le numéro de port correspondant à celui sur lequel écoute votre serveur backend Java.

# Installation

1. Cloner le dépot GitHub  
   git clone [git@github.com](mailto:git@github.com):mxmvncnt/TP1-objets-connectes.git  
   cd TP1-objets-connectes/web
2. Installer angular cli

npm install -g @angular/cli

1. Installer les dépendances necessaires

npm install

1. Mettre en marche l’application web

ng serve