BURGUBURU Côme
CHARBON Hugues
JORNET Benjamin

Projet Majeure :

Ambiance Show Room

http://ambiance.herokuapp.com/

Jacques SARAYDARYAN

CPE Lyon Projet 5ETI

TABLE DES MATIERES :

I] Présentation du projet

1.1) Description du projet

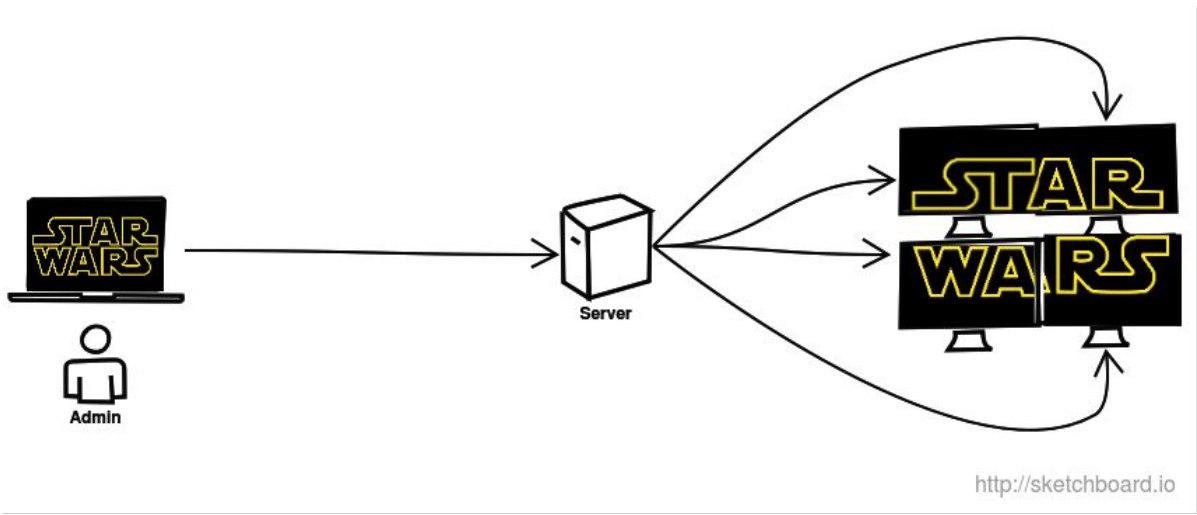
Ambiance Show Room est une application web qui permet de visualiser des flux sur différents écrans. Ces flux peuvent être des images, vidéos, Facebook ou Twitter.

La communication se réalise entre une page d'administration et ses watchers via des sockets. L'utilisateur sélectionne la disposition des écrans via sa page d'admin, et peut choisir quel flux envoyer sur chacun d'eux.

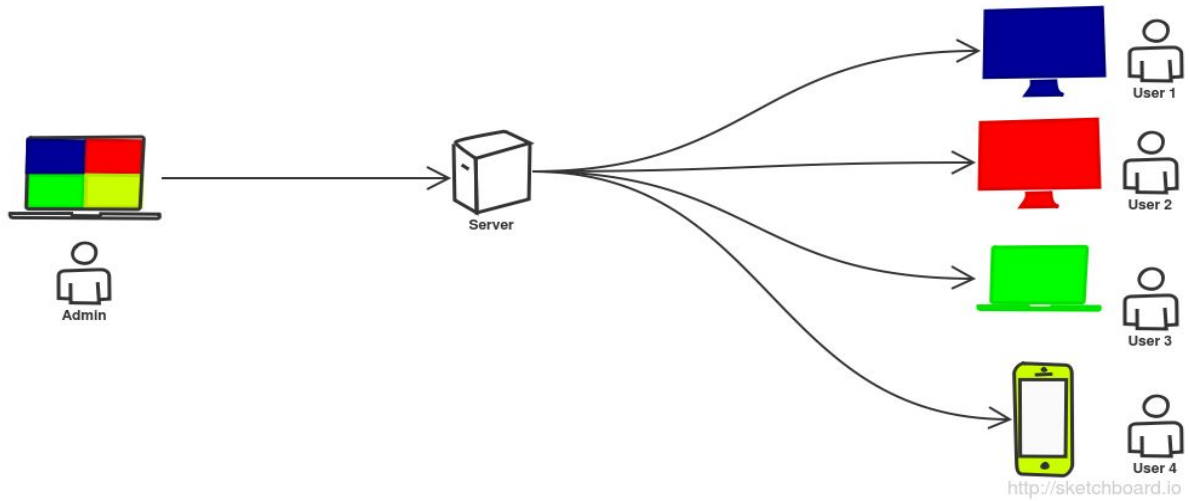
Cette application est aussi compatible en mobile multi-plateforme. Cela signifie qu'il est aussi possible d'envoyer des flux vers des mobiles, ou de gérer les écrans en tant qu'administrateur.

Il y a deux façons d'utiliser cette application :

Première utilisation : Afficher **un même flux** sur plusieurs écrans. Donc, ce flux sera visuellement découpé selon le nombre d'écrans, ainsi que de leur disposition des uns par rapport aux autres. Une grille sera à disposition de l'utilisateur avec les différents watchers, qui correspondront chacun d'eux à un écran.



Seconde utilisation : Afficher **plusieurs flux** sur plusieurs écrans. L'utilisateur, via sa page d'administrateur, va choisir quels flux envoyer sur les écrans. Il est donc possible de diffuser différents flux sur plusieurs écrans.



1.2) Répartition du travail dans le groupe

Nous avons décidé en début de projet de diviser le projet en 9 étapes et de nous les répartir entre les différents membres du groupe. Nous avons utilisé l'outil Git pour une meilleure synchronisation de notre code.

La répartition du travail dans le groupe a été la suivante :

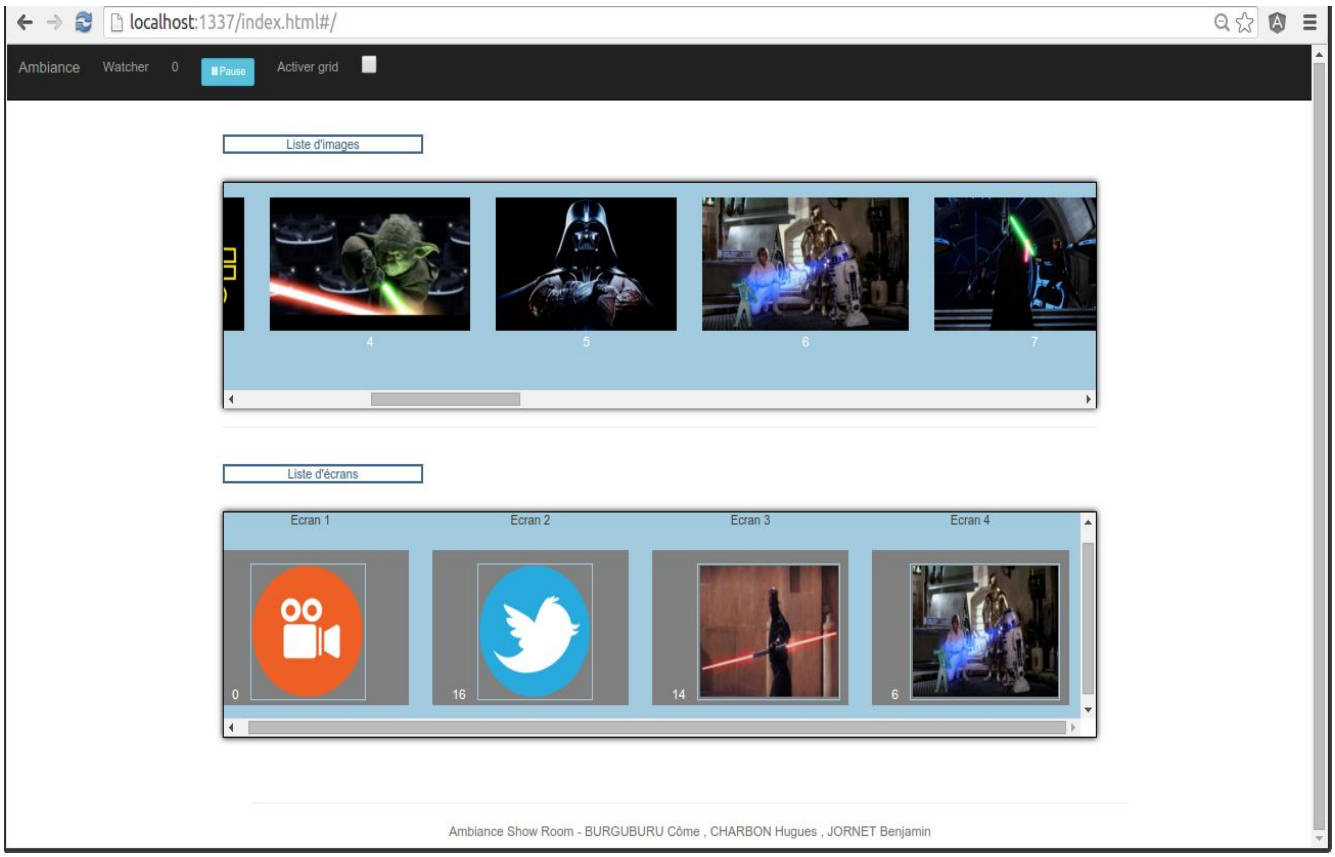
N° Step	Description de la tâche	Élève
étape 1	Identifier + lister des watchers	Côme + Hugues
étape 2	Transférer une image sur le watcher de notre choix	Côme + Hugues
étape 3	Diffuser une image découpée sur plusieurs écrans	Benjamin
étape 4	Configurer la grille des écrans depuis l'administrateur	Hugues + Benjamin
étape 5	Pouvoir transmettre des flux Facebook et Twitter	Benjamin
étape 6	Pouvoir transférer une vidéo	Côme
étape 7	Découper une vidéo	Benjamin
Step 8	Synchroniser la lecture d'une vidéo	Côme
Step 9	Avoir une version plateforme mobile	Côme (Android)+ Hugues(Cordova)

II] Spécifications fonctionnelles et techniques

2.1) La page administrateur

La page d'administrateur permet de re-diriger les flux vers les différents écrans. Comme cela a été décrit précédemment, il y a deux façons d'utiliser cette application :

1. Dans le cas où nous diffusons différents flux vers plusieurs écrans, nous avons l'interface administrateur suivante :

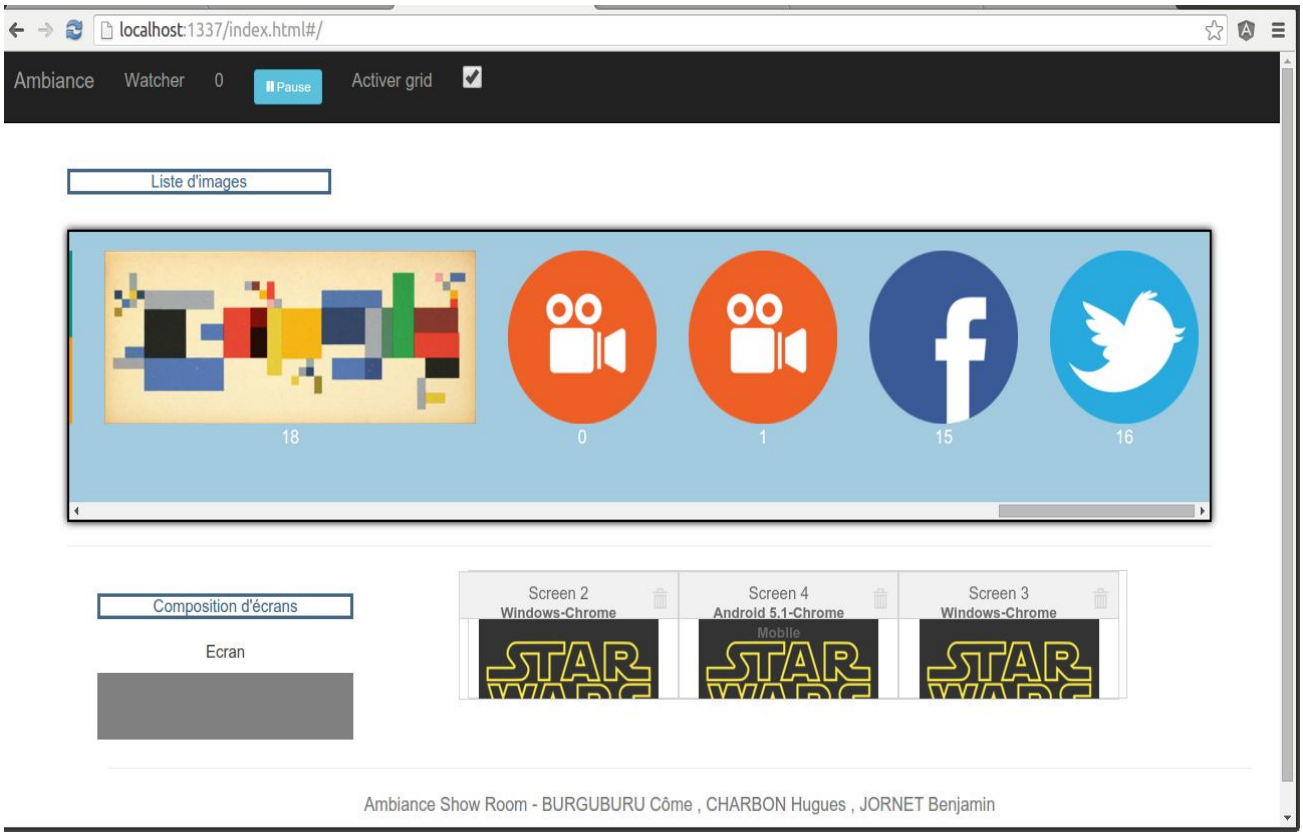


Dans le header, nous pouvons créer un watcher dans un nouvel onglet en cliquant sur ce bouton. Chaque client est identifié par un numéro qui apparaît dans la barre de navigation.

Le bouton “pause” permet de mettre sur pause toutes les vidéos. Nous avons la possibilité de passer à l’autre mode d’administration en cochant la case “Activer grid”.

Pour afficher un flux sur un écran, il faut glisser et déposer une image de la liste d’images vers un écran, qui est identifié de façon unique grâce à un numéro de watcher. Les images de la vidéo ou du logo Twitter, par exemple, sont des représentations de flux.

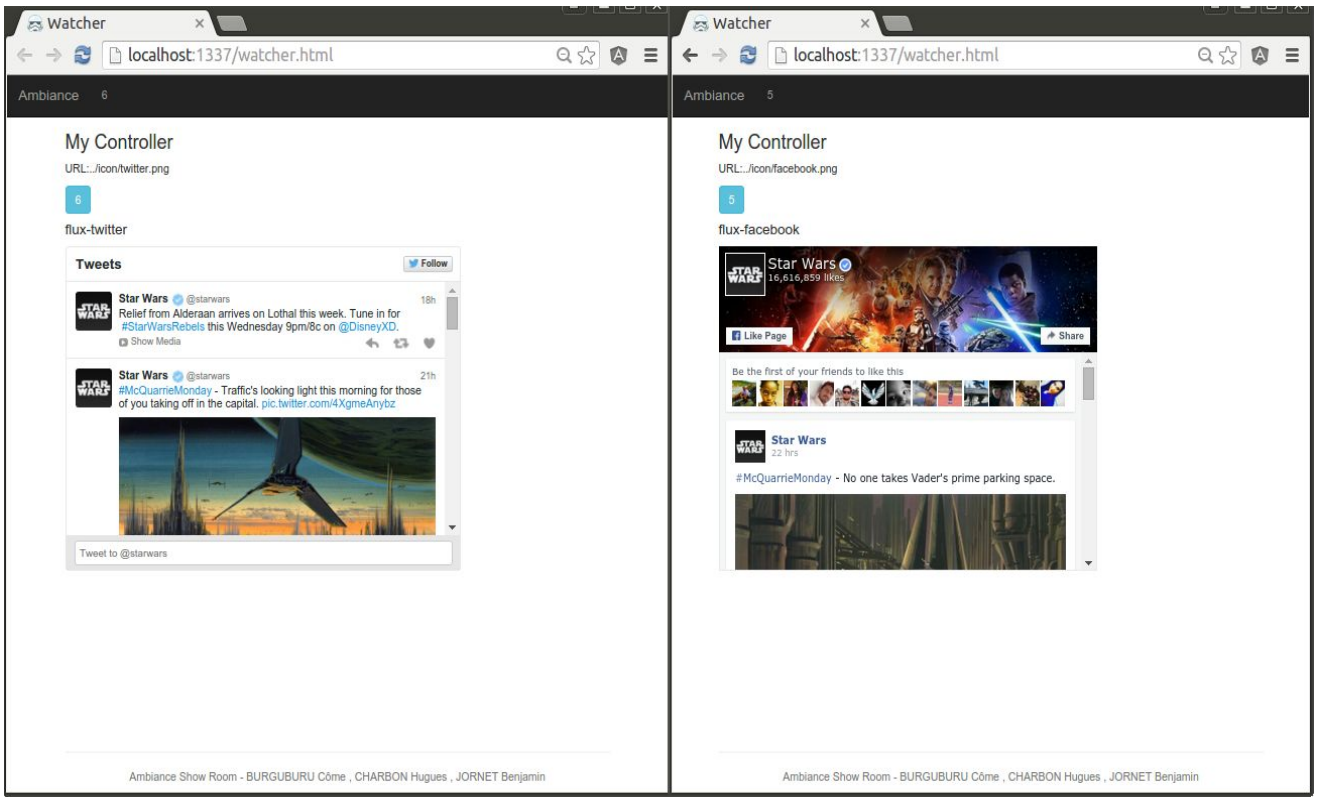
2. Dans le second écran où nous diffusons un même flux sur différents écrans, nous avons l’interface administrateur suivante :



Dans cette configuration, il suffit de glisser et de déposer une image de la “Liste d’images” dans “Composition d’écrans”. Ensuite les dimensions du flux s’adaptent en fonction du nombre de widgets, ainsi que de leurs dispositions des uns par rapport aux autres. Ici, dans notre exemple, l’utilisateur a ouvert 3 watchers, et le flux s’étalera à l’horizontale sur 3 écrans.

2.2) La page watcher

La page watcher permet l'affichage d'un flux sur un écran. Le flux peut être sous différents types : image, vidéo, Facebook ou Twitter. Les dimensions du flux s'adaptent si l'utilisateur souhaite afficher une image ou une vidéo. Nous avons volontairement empêcher ce redimensionnement pour les flux Facebook et Twitter pour des raisons d'ergonomie et d'affichage. Soit l'affichage ci-dessous de flux Facebook et Twitter :



Dans le cas où l'utilisateur souhaite afficher une image ou une vidéo sur plusieurs écrans, le flux est émis vers tous les watchers en question, et des valeurs d'attributs CSS (top, left, width et height) sont aussi envoyés pour déplacer les flux vers les écrans. Cela permet ainsi d'obtenir une impression de "découpage" des flux.

Dans le cas où l'utilisateur souhaite afficher plusieurs flux sur plusieurs écrans, une API fullscreen permet d'afficher le flux en grand écran en double cliquant sur l'image.

Enfin, le rôle du watcher peut se résumer à l'interprétation des données provenant du serveur.

2.3) La communication par socket

La communication entre le serveur et les clients web (AngularJS) a été établie grâce a la librairie Socket.IO. Le choix de cette technologie se justifie par un passage au temps réel et une communication bi-directionnelle. Cela permet une très grande réactivité de l'interface, par une mise à jour en tant réel des informations. De plus cette librairie implémente la technologie heart-beat, qui permet de détecter les clients qui ne sont plus connectés au serveur.

2.4) La synchronisation vidéo

L'affichage des vidéos repose sur la directive ng-media qui génère les balises sources. L'état lecture / pause des vidéos est synchronisé entre les watchers lorsqu'un utilisateur met la vidéo en pause, le serveur est notifié et transmet cet état aux autres watchers pour qu'il se synchronise. Depuis la page administrateur, il est possible de contrôler cet état mais également de choisir un état initial lors de la diffusion d'une vidéo.

2.5) La compatibilité mobile

La compatibilité mobile à été abordée de deux façons différentes.

La première solution s'appuie sur Cordova et conserve toute la logique métier du site. Seule l'interface à été retravaillée afin d'être adaptée à la résolution d'écran d'un smartphone. Cette approche possède malgré tout une faiblesse : l'utilisation de requêtes Ajax n'est pas possible car la webview affiche le site en local.

Pour pallier à cette faiblesse, une version native Android a aussi été développée. En plus de la version JAVA de la librairie socket.io-client, l'application utilise deux librairies de l'entreprise Square. La première OkHttp permet de faire des requêtes asynchrones vers le serveur distant, afin de récupérer la liste des images disponibles sur le serveur au format JSON. La deuxième librairie est Picasso, elle permet de charger et de mettre en cache une image dans une webview à partir d'une URL.

Dans les deux cas, nous nous sommes efforcés d'avoir une interface responsive.

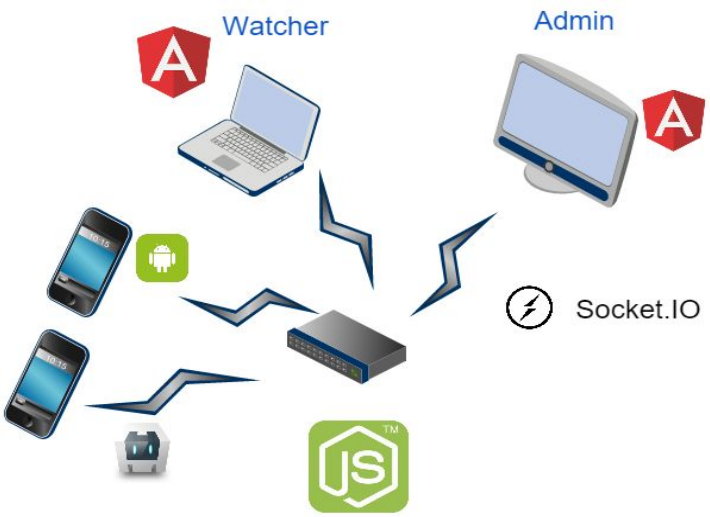
Dans les applications mobiles les scokets sont relié à l'URL de la version en ligne ainsi tous les watchers et les administrateurs sont relié au même serveur d'application.

Cette mise en ligne de l'application sur la plate-forme Herokuapp nous dispense de faire tourné une version local lors de la démonstration et de devoir spécifié l'IP du serveur à chaque démarrage de l'application mobile.

III] Architectures techniques et choix technologiques

3.1) Architecture technique globale

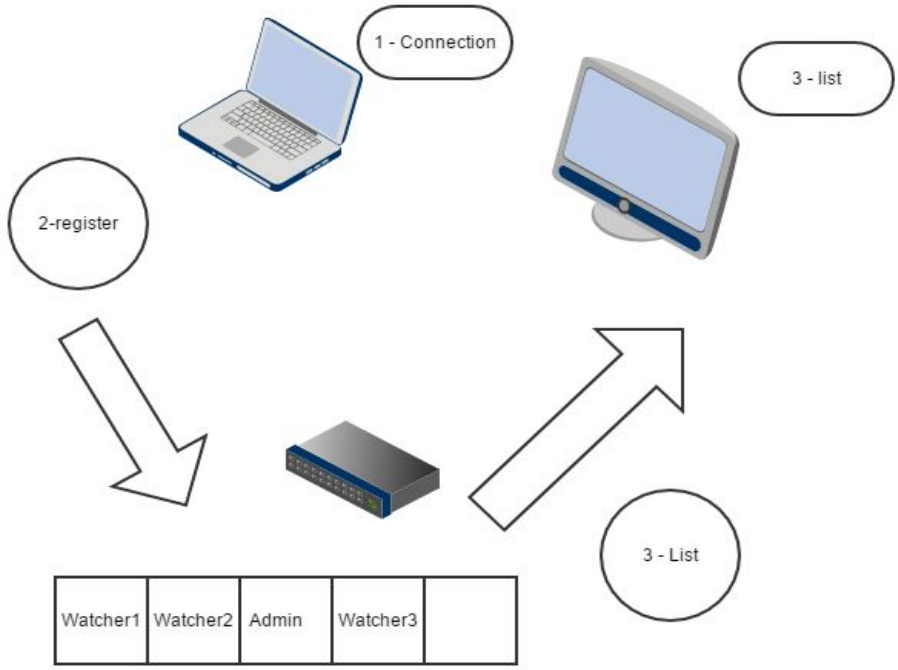
L'utilisateur utilise l'interface d'administrateur, qui permet de dispatcher les flux sur les différents écrans. En pratique, les informations sont envoyées par les sockets (en utilisant Socket.io) vers le serveur Node.js, qui diffuse les flux vers les watchers. Ces derniers peuvent être sous différents types de device (ordinateur, mobile). Nous développerons les choix technologiques dans les parties suivantes.



3.2) Choix technologiques back-end

La partie back-end du projet utilise la technologie NodeJS et Express.

Cette technologie permet une mise en place rapide et permet d'exploiter facilement les sockets, essentielles pour la transmission de flux.



Lors de la connexion d'un watcher, le client émet un évènement "register" pour envoyer au serveur ces propriétés (type de périphérique, taille, userAgent) . La socket est stockée dans un tableau et sa place dans celui-ci servira à l'identifier. Le serveur enverra dans un premier temps un évènement "identification" à ce client, pour lui communiquer son identifiant de watcher. Dans un second temps, l'évènement "list" sera envoyé à tous les administrateurs, avec pour arguments la liste des watchers pour que la liste des clients disponibles soit mise à jour.

3.3) Choix technologiques front-end

En ce qui concerne les choix technologiques front-end, nous avons choisi : HTML5, Javascript et notamment Angular.JS.

Nous avions besoin d'organiser notre développement côté client. Nous avons choisi d'utiliser le framework AngularJS pour concevoir notre application car il possède un certain nombres d'avantages:

- AngularJS utilise le data-binding bidirectionnel, c'est à dire que des modifications dans le modèle sont automatiquement mises à jour dans la vue, et inversement.

- AngularJS utilise des directives qui permettent de manipuler proprement le DOM, qui facilitent l'ajout ou la modification de composants au sein de la page HTML et qui rendent le code extensible et modulable.

- AngularJS utilise l'injection de dépendances, en effet on peut ajouter des services aux contrôleurs pour qu'ils puissent les utiliser au sein de leurs propres fonctionnalités. Cette méthode permet de mieux structurer le front-end de notre application et permet à plusieurs contrôleurs d'appeler le même service.

Pour réaliser la grille, nous avons eu recours à la librairie gridster.js qui permet de gérer les widgets de la grille. Ainsi, en fonction du nombre de widgets, nous modifions les dimensions de l'image grâce à des éléments CSS.

Nous avons aussi utilisé la technologie Angular.js en respectant le modèle Application-Controller-Service. Le controller gère les évènements utilisateur, tandis que les services gèrent la communication des sockets. Par conséquent, nous avons un controller d'évènements utilisateur, qui, couplé à un service, écoute la socket. Nous avons notamment utilisé la directive ng-draggable pour pouvoir utiliser le drap&drop des flux vers les écrans.

De plus, nous avons également utilisé le framework bootstrap pour créer des outils de navigation et rendre notre application responsive.

IV] Tests et périmètre fonctionnel

4.1) Tableau récapitulatif des tests réalisés

Afin de recenser l'intégrité fonctionnelle de l'application, nous avons effectué différents tests afin de pouvoir évaluer ces derniers :

Fonctionnalité	Mode	Périmètre
Envoie d'une image	Simple	1 client
	Grille	3 clients
Envoie d'une vidéo	Simple	1 client
	Grille	2 clients

Un script NodeJS nous a permis de tester facilement le bon fonctionnement de notre algorithme de découpage d'image, en appliquant différentes configurations d'écran en entrée.

4.2) Périmètre fonctionnel

L'application web permet de visualiser en temps réel les différents watchers connectés à l'application. Chaque watcher est capable de recevoir et d'interpréter chaque type de flux, de plus de la lecture du contenu vidéo synchronisée.

Cependant le découpage des images ne prend pas en compte la taille de chaque écran mais choisit par convention la largeur de la première ligne d'écran.

V] Conclusions

Nous avons pu aborder dans ce projet une grande variété de technologies de développement Web : HTML5, JAVASCRIPT, NodeJS, Socket.IO, AngularJS, Android, Cordova ...

Nous avons pu mettre nos connaissances acquises en enseignement de majeure en application dans un cas concret.

Cette mise en pratique de nos connaissances nous a permis de relever la pertinence des différentes technologies, que nous avons utilisées en fonction des utilisations.

Nous sommes aussi conscients des perspectives d'amélioration de l'application comme par exemple :

1. La possibilité de déployer des jeux en ligne via les watchers,
2. La récupération de flux streaming,
3. L'optimisation du front-end,
4. L'identification des utilisateurs via des comptes.

VI] Bibliographie et Webographie

Sites Web :

https://github.com/

https://angularjs.org/

https://nodejs.org/en/

https://developers.facebook.com/

https://dev.twitter.com/

http://developer.android.com/index.html

http://www.w3schools.com/

http://stackoverflow.com/

http://gridster.net/

http://socket.io/

http://getbootstrap.com/

Images : Star Wars