

Projet individuel
PJI

Compte rendu projet final
Développement d'une WebApp

Application :
MEDFERCAST



Binôme :
LAFOLIE Anthony
BRICOUT Gaël

Encadrant:
LE PALLEC Xavier

INTRODUCTION	3
Présentation du projet	3
Cahier des charges	3
Modélisation de l'application	4
MANUEL D'UTILISATION	5
Comment accéder à l'application ?	5
Comment l'utiliser ?	5
Composants/choix de l'application	6
BILAN ET PERSPECTIVES	8

INTRODUCTION

Présentation du projet

Dans le but de créer une application qui dépasse la limite d'un seul écran ou d'un système d'exploitation, nous voulions créer un projet qui nous permettrait d'étudier les manières de connecter des clients entre eux, et de faire en sorte qu'une synchronisation s'établisse entre eux.

Nous avons donc créé une application Web, de type PWA (Progressive Web App), d'une part car c'est la manière la plus simple d'interconnecter des appareils numériques, et d'une autre part car beaucoup de problèmes de connexion (tels que le pare-feu, par exemple) ne seront pas rencontrés, contrairement à une application lourde.

Ensuite, nous devons trouver une idée qui nous permettrait d'étudier cette relation entre les utilisateurs, qui utiliseraient la même application.

Après une étude d'idées, de brainstorming, nous avons eu l'idée de créer une application susceptible d'afficher du contenu de type média à différents utilisateurs. De plus, l'application permettrait de déplacer l'image, et de zoomer (si l'utilisateur souhaite montrer une partie précise de l'image) qui serait synchronisée avec tous les autres utilisateurs, peu importe leur appareil numérique (smartphone, tablette, ordinateur, android TV).

Cahier des charges

Contexte : Créer une application web progressive susceptible de connecter des clients et de synchroniser le partage de médias.

Objectif : Étudier les outils, les technologies et les possibilités afin de réaliser ces interconnexions.

Historique : Première version réalisée il y a 2 mois.

Contraintes :

- Il faut pouvoir envoyer des photos, et faire en sorte qu'elles soient visibles de manière correcte par les autres clients.
- Le zoom d'un client doit respecter le ratio et la taille chez les autres clients
- Le déplacement d'une image d'un client doit déplacer l'image aussi pour les autres clients, en conservant les ratios.
- L'application doit être utilisable depuis n'importe quel appareil numérique connecté.

Organisation :

Afin de mener à bien le projet, nous avons réalisé une répartition des tâches et des objectifs en fonction du temps.

- 14/04/21 : Serveur programmé en local
- 29/04/21 : Envoi des photos entre plusieurs pages internet différentes fonctionnel en local

- 15/05/21 : Envoie de médias entre plusieurs client déployé
(*RETARD* : RÉALISE LE 24/05/21 à cause des examens et de gros problèmes avec Websocket)
- 20/05/21 : Réalisation de “salles” afin de faire en sorte qu’un utilisateur envoie seulement un média aux personnes présentes dans la salle
- 23/05/21 : Zoom et déplacement par un client zoom et déplace chez les clients présents dans la salle
(*RETARD* : RÉALISE LE 31/05/21 à cause des problèmes de calculs de via JavaScript/TypeScript)
- 31/05/21 : Mots de passes pour les salles déjà existantes
(*RETARD* : RÉALISE LE 03/06/21 à cause des précédents retards).

Modélisation de l'application

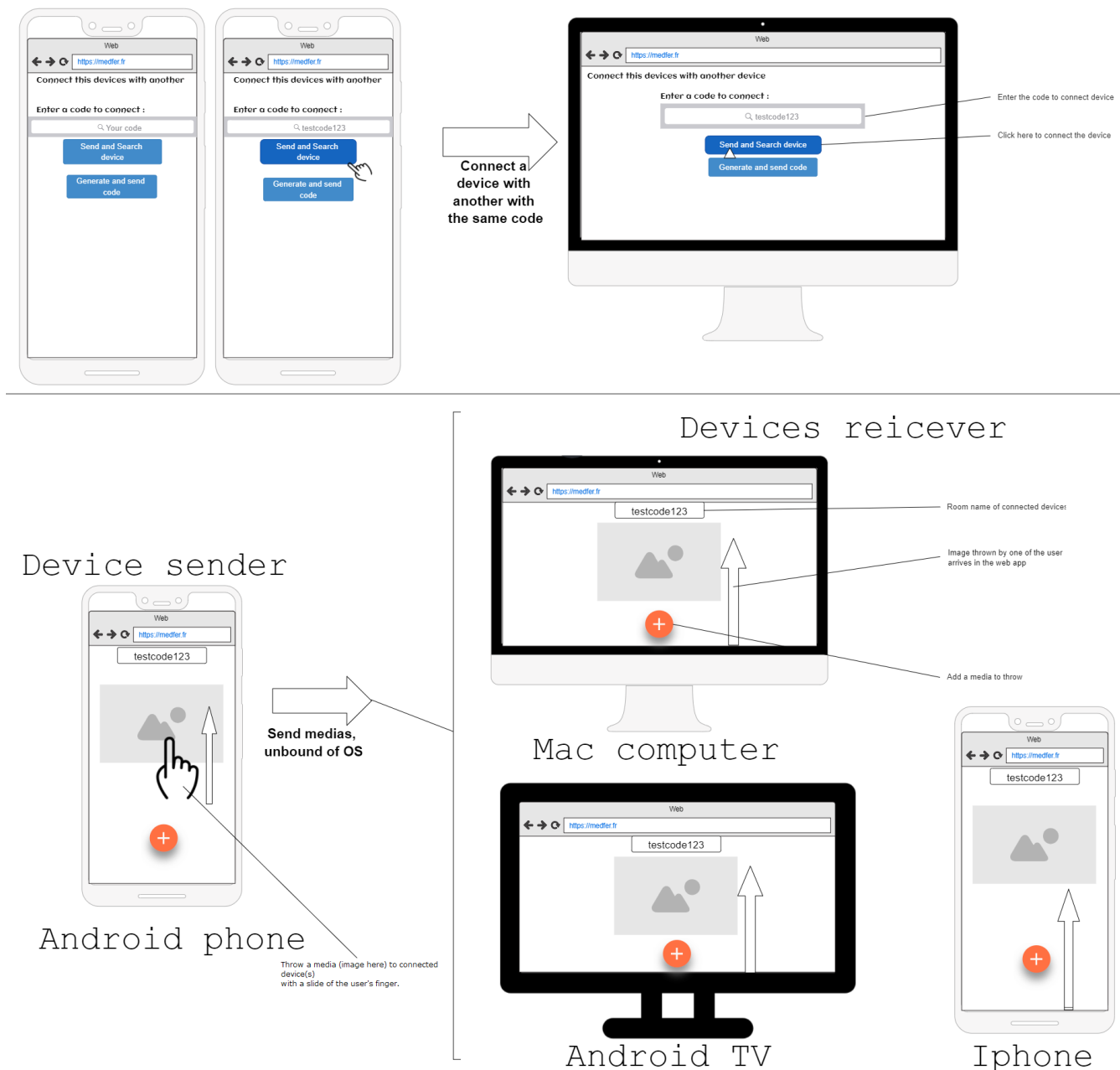


Figure 1 : Modélisation de l'application

MANUEL D'UTILISATION

Comment accéder à l'application ?

L'application est déployée, et donc accessible sur internet. Le serveur est aussi stocké sur Heroku, qui était une solution simple afin de déployer de manière à exécuter des tests. Le lien est : <https://medfercast.herokuapp.com/>

Comment l'utiliser ?

Une fois le site lancé, une page d'accueil est affichée.

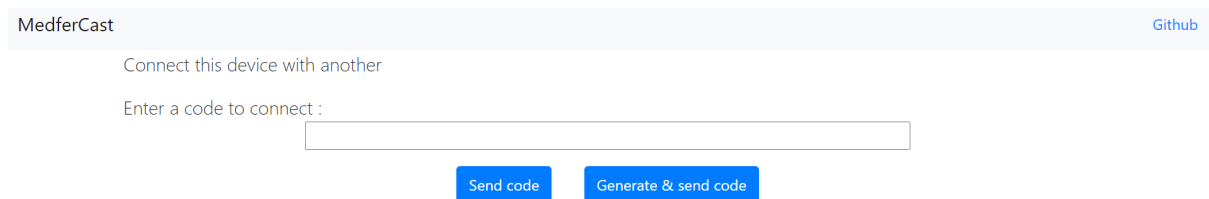


Figure 2: page d'accueil de MerdferCast.

Un champ de texte y est affiché, dans lequel il faut y indiquer le code (le nom de la salle) à rejoindre. Deux cas sont possibles : 1° créer sa salle si le nom n'existe pas ou 2° rejoindre une salle avec un nom déjà existant.

Une fois le code rédigé, appuyer sur le bouton *send code* affichera une page qui vous demandera d'y indiquer un mot de passe (soit de la salle déjà existante, soit de la salle que vous allez créer).

Sinon, le bouton *Generate & send code* va vous créer un code de 4 caractères qui n'est pas encore attribué. Une page va alors s'ouvrir afin d'y indiquer le mot de passe que vous voulez mettre dans cette salle.

Ensuite remplir le code demandé, et appuyer sur *Add password* qui ouvrira la salle, et connectera l'utilisateur.

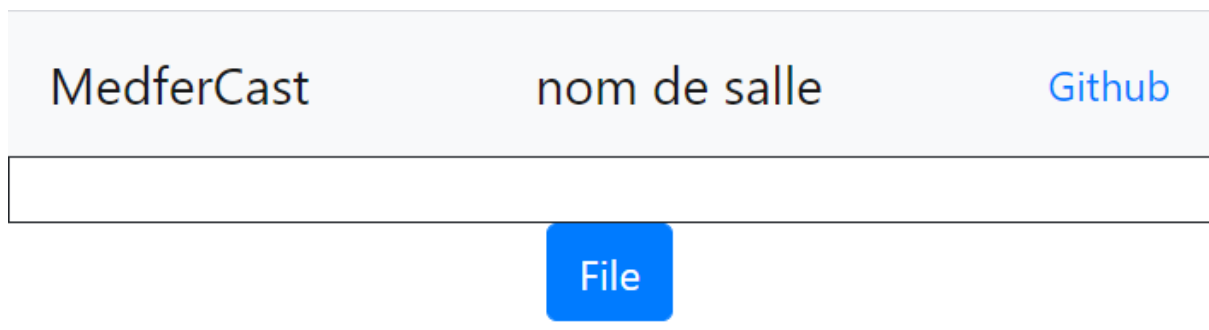
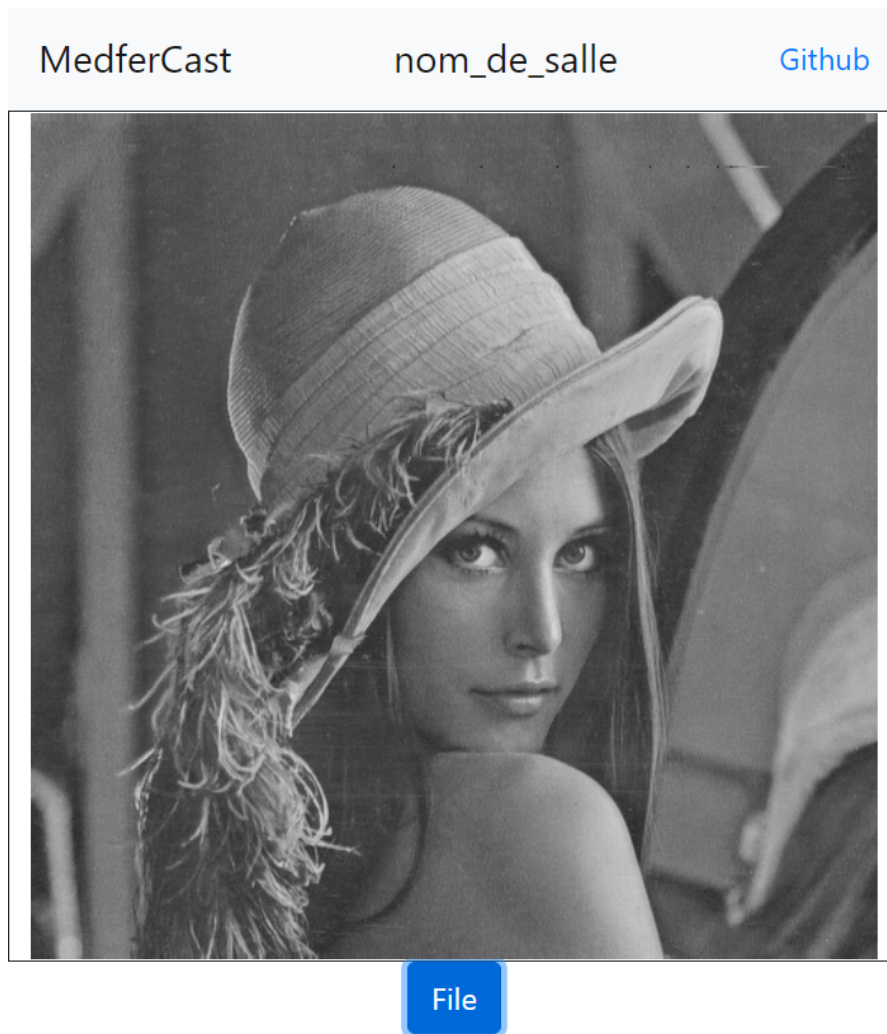


Figure 3 : utilisateur connecté à la salle <nom de salle>

Enfin, cliquer sur le bouton *File* va permettre à l'utilisateur de choisir dans ses données une photo, et l'enverra aux autres clients connectés à la salle.



**Figure 4 : Média
envoyé depuis
l'application aux
utilisateurs
présents dans la
salle
<nom_de_salle>**

Selon l'appareil avec lequel l'utilisateur se connecte, il sera possible de déplacer et de zoomer l'image (respectivement au touché pour le smartphone ; au clic pour la souris, et au pincement de doigts pour un smartphone ; à la molette pour la souris), qui va synchroniser ce changement aux autres utilisateurs de la salle.

Composants/choix de l'application

Tout d'abord, il fallait étudier les technologies que nous allions utiliser afin de créer ce projet. Nous avons observé ce qui se fait le plus dans les entreprises, et les technologies les plus poussées et mises au premier plan récemment. Nous avons donc choisi d'utiliser Angular, accompagné de Node.

- Angular CLI: 9.0.3 : partie Front de l'application, et s'occupe du côté client.
- Node: 12.14.1 : le serveur qui permet, lors d'un envoi d'un client d'une image, de redistribuer cette image aux différentes salles.

Ensuite, nous devons étudier quelle bibliothèque nous allions utiliser afin de connecter les clients. Après des recherches et des questionnements sur nos réels besoins, nous avons éliminé le streaming direct (notamment proposé par streamdata.io par exemple) car cela

aurait demandé beaucoup plus de temps, avec la création d'un serveur plus puissant et l'obligation de programmer une API. Nous pensions que nos besoins n'étaient pas si importants.

Une autre possibilité aurait été d'utiliser le style architectural REST, et plus précisément avec la contrainte HETEOAS qui permet en fait de mettre les hypermédias en tant que moteur central de l'application. Cependant, ici aussi, une obligation de mettre en place une API était présente.

Nous nous sommes donc orientés vers les sockets, et plus précisément ce qu'il se fait le plus de nos jours, les websockets. La première version de l'application était gérée à ce niveau par la bibliothèque WS, qui ne sont autres que des websockets. Nous avons eu énormément de problèmes (qui a impliqué du retard) avec cette bibliothèque, qui nous a donc fait changer notre fusils d'épaules, et passer à notre deuxième choix Socket.io

→ Socket.io: 2.1.1 : bibliothèque permettant d'utiliser les websockets, qui donne à l'application la possibilité de recevoir, d'envoyer des sockets et d'orienter les flux de données.

Son point fort est que cela permet la communication bidirectionnelle et basée sur les événements.

Son point faible est que lorsqu'une image est lourde (>500Ko), il est impossible de l'envoyer sans la découper

BILAN ET PERSPECTIVES

Notre application est fonctionnelle et permet d'exécuter notre premier souhait. Cependant, selon nous, il y a encore beaucoup de détails à régler et des choses à ajouter. Par exemple, ajouter un nom d'utilisateur pour savoir qui est dans la salle, une possibilité de scinder la zone de réception des photos, la possibilité de caster des vidéos, un serveur plus puissant qui permettrait à l'utilisateur de télécharger les photos envoyées, etc.... Ce sont des ajouts qui pourraient potentiellement être fait par la suite, en dehors de notre projet PJI, car cela nous pousserait à découvrir encore plus ces technologies.

Ce projet nous a permis d'en apprendre plus sur beaucoup de domaines des E-services. Notamment sur l'utilisation d'Angular, node, et surtout les technologies présentes afin de construire des applications réparties. Nous avons pu perfectionner nos connaissances côté serveur (Back-end) et sur les Framework Front-end, et mettre à exécution des cours que nous avons suivis cette année.

Nous sommes satisfait de ce dont nous avons réalisé et remercions Monsieur Le Pallec pour son accompagnement et son encadrement dans notre projet.