



Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement 2021-2022

Solution de capture et composition temps-réel de flux video sur mobile

Inside



POLYTECH®
TOURS

Entreprise

Polytech



Tuteur entreprise

Sylvain FABRE

Étudiant

Adrien PREVOST (DI5)

Tuteur académique

Mathieu DELALANDRE

3 avril 2022

Liste des intervenants

Entreprise

Polytech
64 avenue Jean Portalis
37200 Tours, France
polytech.univ-tours.fr



Nom	Email	Qualité
Adrien PREVOST	adrien.prevost@etu.univ-tours.fr	Étudiant DI5
Mathieu DELALANDRE	mathieu.delalandre@univ-tours.fr	Tuteur académique, Département Informatique
Sylvain FABRE	contact@sylvainfabre.com	Tuteur entreprise



Avertissement

Ce document a été rédigé par Adrien PREVOST susnommé l'auteur.

L'entreprise Polytech est représentée par Sylvain FABRE susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Mathieu DELALANDRE susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Adrien PREVOST, *Solution de capture et composition temps-réel de flux video sur mobile: Inside*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
  author={PREVOST, Adrien},
  title={Solution de capture et composition temps-réel de flux video sur mobile: Inside},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iv
1 Introduction	1
1 Acteurs, enjeux et contexte	1
2 Objectifs	1
3 Bases méthodologiques	1
2 Description générale	3
1 Environnement du projet	3
2 Caractéristiques des utilisateurs	3
3 Fonctionnalités du système	3
4 Structure générale du système	4
3 État de l'art / Veille technologique	5
1 Étude des projets précédents	5
2 Technologies disponibles	5
4 Analyse et conception	7
1 Analyse	7
1.1 Hypothèses utilisées	7

1.2	Décisions	7
2	Modélisation proposée.....	8
3	Interface	8
4	Premiers tests.....	8
5	Mise en oeuvre	12
1	Introduction	12
2	Le serveur HTTP.....	12
2.1	Modèle de données.....	13
2.2	URLs.....	13
2.3	Connexion à un concert	14
3	Le serveur Socket.IO.....	14
3.1	Exemple	14
3.2	Messages	15
4	Fusion des vidéos	15
4.1	FFMPEG	15
4.2	Découpage des vidéos	16
4.3	Concaténation des vidéos.....	16
4.4	Transfert de l'audio	16
5	Diffusion sur Facebook.....	17
6	Application mobile.....	18
6.1	Introduction	18
6.2	Séparation des éléments.....	18
6.3	Librairie de communication.....	19
6.4	Caméra et variables globales	19
7	Bugs et améliorations futures.....	20
7.1	Vidéo renversée.....	20
7.2	Chevauchement des checkpoints.....	20
7.3	Facebook	20
6	Bilan et conclusion	22
1	Bilan du semestre 9	22
2	Bilan du semestre 10.....	22
	Annexes	23
A	Planification, gestion de projet	24
1	Evolution du projet	24

B	Cahier de Spécifications	25
1	Contexte de la réalisation	25
1.1	Objectifs.....	25
1.2	Hypothèses	25
1.3	Bases méthodologiques	25
2	Description générale.....	26
2.1	Environnement du projet	26
2.2	Caractéristiques des utilisateurs	26
2.3	Fonctionnalités du système	26
3	Description des interfaces externes du logiciel.....	27
3.1	Interfaces matériel/logiciel	27
3.2	Interfaces homme/machine	27
3.3	Interfaces logiciel/logiciel	29
4	Spécifications fonctionnelles	29
5	Spécifications non fonctionnelles	29
5.1	Contraintes de développement et conception	29
5.2	Capacité	30
C	Document d'installation	31
1	Serveur	31
1.1	Node.js	31
1.2	FFMPEG	31
1.3	Page facebook.....	31
2	Application mobile.....	31
D	Document d'utilisation	32
1	API	32
2	Application Mobile	32

Table des figures

1	Introduction	
1.1	Exemple de vision d'un concert Inside.....	1
2	Description générale	
2.1	Schéma de représentation du système.....	4
4	Analyse et conception	
4.1	Diagramme de séquence du système.....	8
4.2	Page d'accueil de l'application	9
4.3	Vue horizontale, concert créé sur le téléphone principal	9
4.4	Vue verticale, concert créé sur le téléphone principal	10
4.5	Vue de la page facebook.....	11
5	Mise en oeuvre	
5.1	Schéma de l'organisation logique de l'application	12
5.2	Vue juste avant la connexion.....	18
5.3	Vue menu principal.....	19
5.4	Vue durant un concert	19
A	Planification, gestion de projet	
A.1	Le diagramme de Gantt Initial.....	24
A.2	Le diagramme de Gantt Final	24

B Cahier de Spécifications

B.1 Schéma synthétique du système	26
B.2 Page d'accueil de l'application	27
B.3 Vue horizontale, concert créé sur le téléphone principal	28
B.4 Vue verticale, concert créé sur le téléphone principal	28
B.5 Diagramme de séquence du système	30

1

Introduction

1 Acteurs, enjeux et contexte

En temps de pandémie, il est difficile d'organiser des concerts, et encore moins avec beaucoup de monde. Sylvain Fabre, musicien professionnel, souhaiterait donc un moyen facile de diffuser les concerts sur internet en direct pour faire profiter un maximum de personnes même à distance.

Le projet Inside cherche à résoudre ce problème en permettant aux musiciens de diffuser leurs concerts sur une plate-forme en ligne.

De plus, Inside propose différents points de vue, à l'aide de caméras portées directement par les musiciens pour des points de vue inédits.



Figure 1.1 – *Exemple de vision d'un concert Inside*

2 Objectifs

L'objectif du projet est de mettre en place ce système de diffusion audio/vidéo sur des réseaux sociaux, d'une façon accessible aux musiciens ainsi qu'aux spectateurs.

3 Bases méthodologiques

Pour mettre à bien ce projet, j'ai mis en place un planning prévisionnel contenant les grandes tâches à réaliser.

J'ai également mis en place un Git pour, dans un premier temps, pouvoir conserver mon travail entre plusieurs machines, et faire du versioning pour le développement du système.

2

Description générale

1 Environnement du projet

Inside est une plate-forme de diffusion de concerts en direct. Plusieurs caméras et un micro sont disposés sur scène, les flux audio et vidéo sont récupérés puis envoyés sur une plate-forme en ligne. L'installation des caméras et micro est faite par le/les artistes présents.

Ce système a déjà été réalisé lors de précédents projets. Des caméras spéciales et un PC devaient être présents sur scène. Le PC récupérait les flux des caméras et les retransmettaient sur un site web en direct. Le site web permettait de changer de vue entre les différentes caméras grâce à une mosaïque.

L'objectif principal du projet est d'adapter ce système en se passant à la fois du PC et des caméras sur scène et du site web.

2 Caractéristiques des utilisateurs

Les premiers utilisateurs du système sont des musiciens non spécialistes techniques. Le système doit donc être facile d'utilisation et transparent pour l'utilisateur. Les musiciens doivent pouvoir utiliser sans connaissances informatiques poussées.

De l'autre côté du système, les utilisateurs finaux sont les spectateurs en ligne, ils doivent pouvoir accéder à la diffusion du concert en direct facilement sur une page d'un réseau social.

3 Fonctionnalités du système

Le système doit être capable de :

- Créer un concert en ligne
- Assigner des téléphones à un concert
- Diffuser un flux vidéo composé des différentes vues des téléphones sur un réseau social

4 Structure générale du système

Le système devra être composé d'un serveur web distant qui viendra récupérer les flux vidéos, les traiter et les poster sur les réseaux sociaux. Le système doit également être composé d'une application mobile, cette application permettra aux musiciens d'accéder au serveur, pour y créer un concert et faire les enregistrements vidéos.

Lorsqu'associée à un concert, l'application proposera une visualisation de la caméra et commencera à enregistrer des vidéos, qui seront envoyées au serveur.

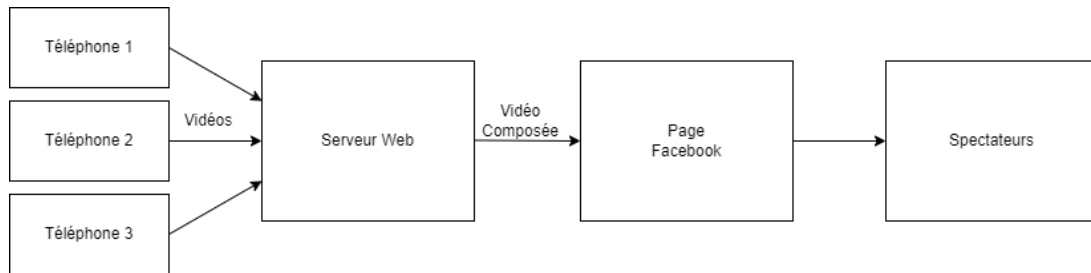


Figure 2.1 – Schéma de représentation du système

3

État de l'art / Veille technologique

1 Étude des projets précédents

Inside a déjà été le fruit de projets passés, j'ai eu accès au rapports et je les ai étudiés pour savoir s'il y avait des éléments que je pouvais récupérer comme brique logicielle pour cette adaptation mobile.

Cette adaptation mobile a déjà également été le sujet d'un stage cet été. La majorité du stage a été passée à étudier les différents réseaux sociaux et la possibilité d'y appliquer ce projet. La conclusion de cette étude a été de choisir Facebook comme réseau social final. Facebook ne permet pas de faire de live via une API, mais permet de poster des vidéos d'une durée maximale de 2h. Les seuls réseaux acceptant de faire des lives avec une API sont Youtube et Twitch qui sont des réseaux spécialisés. Le but étant de pouvoir accéder au concert au milieu de ses informations habituelles, c'est Facebook qui a été retenu, le live est donc simulé par incrémentation successive d'une vidéo postée.

Selon l'étude l'API facebook nécessite une validation de la part de Facebook pour être utilisée complètement, il faudra donc voir si cette validation est possible, sinon il faudra utiliser un outil de simulation de navigateur pour poster "manuellement" la vidéo.

Un projet se concentrait sur l'utilisation de FFMPEG pour le traitement vidéo, cependant, leur but était de réaliser une mosaïque donc tout n'est pas réutilisable, sachant que mon but n'est pas de réaliser une mosaïque, mais une vue successive des différents point de vues.

Les autres projets se basaient sur une diffusion en direct à l'aide de caméra particulières et non d'une application téléphone. Leur code n'est donc pas réutilisable dans ce projet.

2 Technologies disponibles

Le système est découpé en 2 grandes parties, l'application mobile, et le serveur web. Il faut donc trouver une technologie sur laquelle sera basée chaque partie.

J'ai donc pris, comme base de recherche, les technologies que j'ai déjà manipulées. Pour la partie serveur web, j'ai donc regardé la faisabilité du projet avec Node.js. Node permet de créer des serveurs webs légers et portables. Node est capable d'utiliser FFMPEG pour le traitement vidéo et il y a également une librairie pour interagir avec Facebook. Dans le cas où l'API facebook n'est

pas utilisable, il existe une librairie "puppeteer" qui permet de simuler un navigateur chrome sans interface. J'ai donc choisi d'utiliser Node.js pour la partie serveur.

Pour la partie développement mobile, il y a quelques possibilités, Java Android, Flutter, Swift , notamment. Les musiciens peuvent avoir un iPhone ou un android, j'ai donc choisi comme critère le développement cross-platform. C'est donc sur Flutter que je me suis arrêté, ayant déjà de l'expérience avec ce framework et le langage Dart sur lequel il est basé.

Flutter possède des librairies pour contrôler les caméras du téléphone, et des librairies de communication web, il y a donc tout ce qu'il faut pour réaliser le projet.

Il reste à définir les conditions d'hébergement du serveur web. Le serveur doit être disponible en permanence et doit avoir suffisamment de performance pour gérer la gestion des flux vidéos. Cette étude de performance nécessite un système au moins partiellement fonctionnel, elle est donc repoussée au 2ème semestre.

4

Analyse et conception

1 Analyse

1.1 Hypothèses utilisées

L'hypothèse principale est l'utilisation de l'API Facebook. Si l'API est utilisable alors il sera plus simple de l'utiliser via la librairie facebook Javascript. Si elle n'est pas utilisable, il faudra alors simuler un navigateur à l'aide de la librairie puppeteer pour aller poster les vidéos comme si on le faisait manuellement.

A l'heure de rendu du rapport (10/12/2021), le processus de revue de l'application par Facebook est en cours, il est dit que cela peut prendre au moins une semaine sans précisions supplémentaires.

1.2 Décisions

Par rapport à la modélisation, plusieurs décisions ont dû être prises.

Premièrement, la façon dont sont créées les vidéos. Facebook n'accepte pas la création de live, il faut donc créer des vidéos, la question est de savoir si l'application mobile crée les vidéos puis les envoie au serveur ou si le flux vidéo est transmis en permanence entre la caméra et le serveur et le serveur se charge de créer les vidéos à partir du flux vidéo reçu.

La décision a été prise de faire les vidéos directement sur le téléphone pour plusieurs raisons :

- Gérer les problèmes de connexion au téléphone, si l'on perd la connexion à un téléphone, la vidéo étant faite en local, elle ne sera pas altérée, alors que si l'on transfère un flux, la vidéo est perdue.
- Qualité vidéo, sur la même veine que le point précédent, si la connexion se détériore, on peut perdre de la qualité vidéo/audio ou avoir des sauts d'images, ce qui n'est pas le cas si la vidéo est créée en local.
- Simplification de l'utilisation de FFMPEG, ayant de l'expérience avec FFMPEG, gérer des vidéos déjà faites est globalement plus simple que de devoir créer des vidéos à partir d'un flux. De plus, si une connexion est perdue, il faut revoir la façon dont sont faites les vidéos au vol, alors qu'avec des vidéos déjà faites, si le serveur ne reçoit pas vidéo, elle ne passera pas dans la rotation et c'est tout.

Deuxièmement, la gestion du son. Doit-on récupérer le son de chaque téléphone, et changer le son lorsque la vue change ? La réponse qui a été apportée par Sylvain Fabre est non. Un seul téléphone doit transmettre l'audio du concert, celui pourra être connecté à un vrai micro et deviendra le téléphone principal du concert.

Enfin la confidentialité du concert, il ne faut pas que n'importe qui puisse se rajouter au concert vienne se mêler aux prises de vue. Il faudra donc lors de la création d'un concert, donner un nom et un mot de passe au concert, informations qui devront être redonnées lors de l'ajout d'un nouveau téléphone au concert.

Un point qui n'a pas encore été décidé est la conservation de l'entièreté du concert en vidéo. Doit-on conserver uniquement les X dernières minutes du concert, ou le concert en entier ? Le principe du système est de proposer des concerts en live. Conserver la vidéo entière permettrait de revoir le concert depuis le début mais ce n'est pas le principe, donc si la conservation de la vidéo de la vidéo entière crée de la latence par rapport au live, seules les dernières minutes seront conservées.

2 Modélisation proposée

Voici le diagramme de séquence de la modélisation proposée pour ce système. Certains détails sont encore flous, tels que la synchronisation entre les vidéos, mais le concept est là.

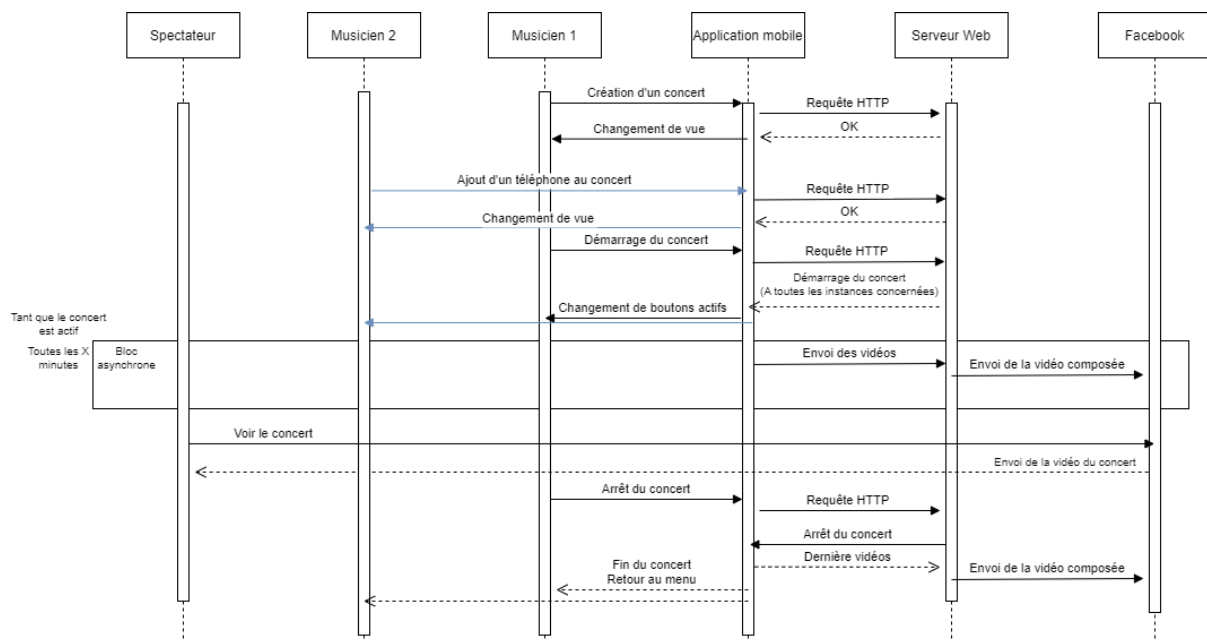


Figure 4.1 – Diagramme de séquence du système

3 Interface

L'interface utilisateur de l'application mobile devra ressembler à quelque chose comme ces schémas ci-après, ils montrent l'interface minimale pour le fonctionnement de l'application. (Figures 4.2, 4.3 , 4.4)

4 Premiers tests

Pour montrer la faisabilité du projet, j'ai fait quelques tests préalables sur l'API Facebook.

Vue page d'accueil

The diagram shows a vertical container with the following elements from top to bottom:

- A rectangular input field labeled "Nom du concert".
- A rectangular input field labeled "Mot de passe".
- A larger rectangular input field labeled "Description".
- A rounded rectangular button labeled "Créer un concert".
- A rounded rectangular button labeled "Rejoindre un concert".

Figure 4.2 – Page d'accueil de l'application

Vue téléphone principal

The diagram shows a horizontal container with the following elements:

- A large rectangular area on the left, outlined in red, labeled "VUE CAMERA" in red text.
- Below the camera view, a horizontal slider labeled "Gestion du zoom" with a black dot at the left end.
- On the right side, there are two rounded rectangular buttons stacked vertically: "Démarrer le concert" (top) and "Terminer le concert" (bottom).

Figure 4.3 – Vue horizontale, concert créé sur le téléphone principal

Pour commencer à utiliser l'API Facebook, il faut créer une application sur le site :

<https://developers.facebook.com/apps/>

Dans les paramètres de l'application, on peut créer une page associée à l'application et également créer un token pour accéder à cette page de manière programmatique.

J'ai ensuite préparé une machine virtuelle Debian 11 pour avoir un environnement similaire au

Vue téléphone principal



Figure 4.4 – Vue verticale, concert créé sur le téléphone principal

futur serveur distant. Et j'ai écrit un programme de test Node.js pour tester la connexion à la page.

```

1  const fs = require('fs');
2  const fbUpload = require('facebook-api-video-upload');
3  //Paramètres de la communication
4  const args = {
5    token: "EAAI4T4qXwOcBAAn2K4Njx42r0e2y",
6    id: "me",
7    stream: fs.createReadStream('WineryDogs_Elevate.mp4'),
8    //path to the video,
9    title: "my video",
10   description: "my description",
11   //you can add any extra fields from the api
12   //all keys except token, id, stream are passed to the final request
13 };
14 //Upload de la vidéo
15 fbUpload(args).then((res) => {
16   console.log('res: ', res);
17   //res: { success: true, video_id: '1838312909759132' }
18 }).catch((e) => {
19   console.error(e);
20 });

```

Le programme utilise une librairie qui a été faite spécialement pour uploader des vidéos. A

l'exécution de ce programme avec le bon token on retrouve bien la vidéo sur la page, après 1min de traitement pour une vidéo de 5min

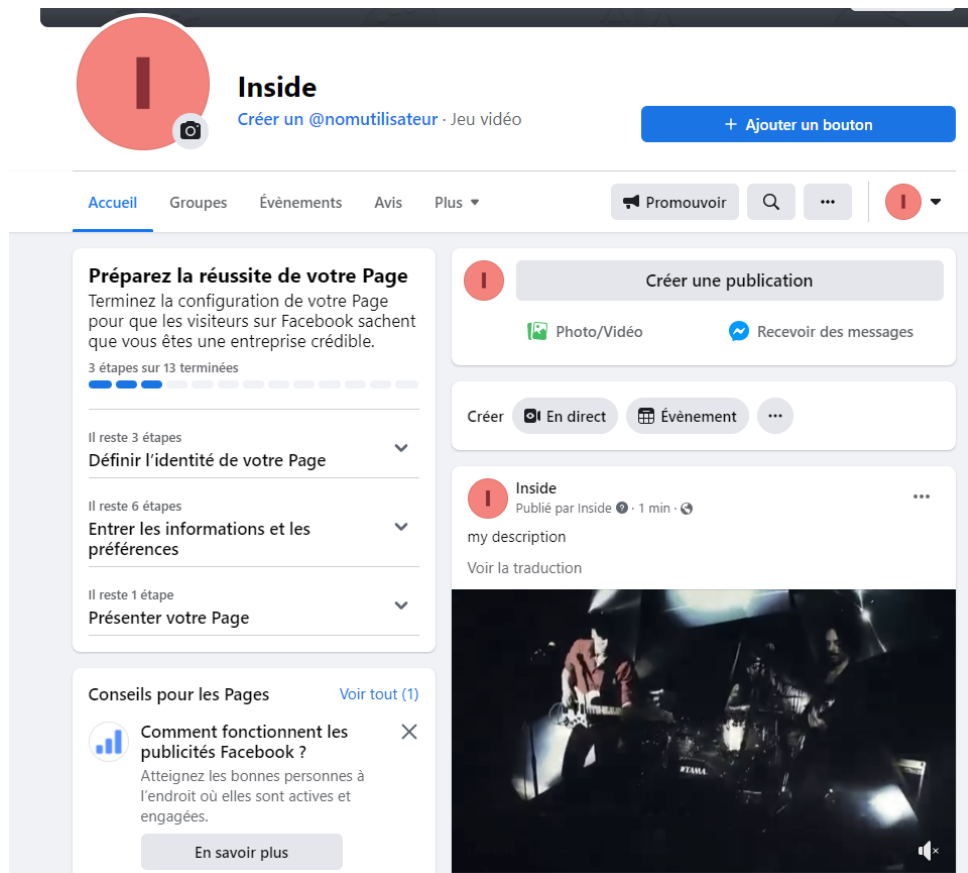


Figure 4.5 – Vue de la page facebook

Malheureusement, les tokens créés manuellement ne durent qu'une heure, et pour des tokens non périssables, il faut passer la revue d'application par Facebook. Mais si cette revue fonctionne, il sera relativement d'intégrer avec la page facebook.

Cependant après, revue par Facebook, ma demande a été refusée. Il faut en fait une application déjà fonctionnelle où il ne manque plus que l'interaction avec Facebook pour que l'utilisation soit validée.

Il serait donc potentiellement possible de refaire une demande une fois l'application complète.

5

Mise en oeuvre

1 Introduction

Le système est donc composé de 2 parties principales : - Le serveur - L'application mobile
On peut voir un schéma synthétique de l'application ci-après.

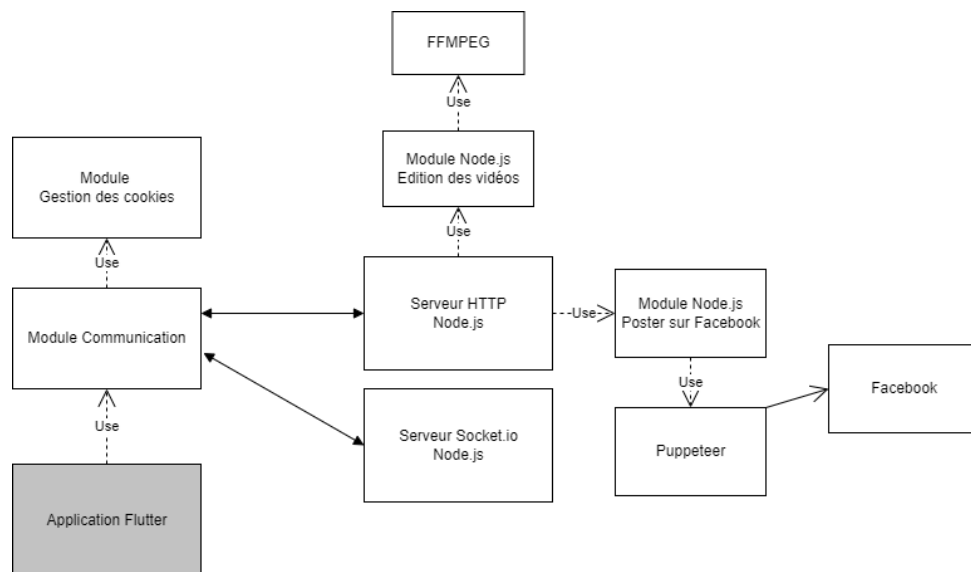


Figure 5.1 – Schéma de l'organisation logique de l'application

Le serveur synchronise les opérations, traite les données et les fait le passe-plats entre les téléphones et Facebook.

2 Le serveur HTTP

Le serveur HTTP permet d'interagir avec le modèle de données de concert.

2.1 Modèle de données

Le serveur gère des concerts. Une liste des concerts en cours est initialisée au démarrage du serveur et est mise à jour lors de création ou d'arrêt de concerts. Un concert est constitué des données suivantes :

```

1 {
2   id: number; //Identifiant unique du concert
3   titre: string; //Titre du concert
4   description: string; //Description du concert
5   password: string; //Mot de passe pour se connecter au concert
6   mainCookie: string; //Cookie donné au créateur du concert
7   //Cookie donné aux autres connectés au concerts
8   sideCookie: string;
9   //Liste des sockets actuellement connectés au concert
10  sockets: Socket [];
11  //Permet de gérer la péremption d'un concert
12  derniereUtilisation: number;
13  // Vidéos uploadées au concert
14  videos: {path:string , audio : boolean}[];
15  enAttenteVideos: boolean; //Le concert attend-t-il des vidéos
16  demarre: boolean; //Le concert est-il démarré
17 }
```

Des URLs sont mises à disposition pour interagir avec ce modèle.

2.2 URLs

Voici une liste des URLs permettant d'interagir avec le modèle de données :

- /api/concert/create : Permet de créer un concert
- /api/concert/join : Permet de rejoindre un concert (récupérer son cookie)
- /api/concert/end : Permet de terminer un concert
- /api/concert/video : Permet d'uploader une vidéo vers le serveur

Pour les détails sur l'utilisation de ces URLs, se référer au fichier "API_README.md" du serveur.

Le serveur est basé sur le framework **Express**. Express permet de créer des serveurs HTTP simplement avec Node.js .

```

1 const express = require( 'express' )
2 const app = express()
3
4 app.get( '/', function (req, res) {
5   res.send( 'Hello World' )
6 })
7
8 app.listen(3000)
```

Ce code montre comment simplement exécuter une fonction à la réception d'un message sur l'URL '/'

Pour certaines fonctions particulières il faut utiliser un middleware. Un middleware vient réaliser une action sur chaque requête arrivante, comme une sorte de prétraitement.

Pour nos besoins ici, on utilise le middleware `fileUpload` pour séparer les fichiers de la requête. On peut ensuite récupérer les fichiers grâce à un simple attribut de la requête.

```

1 app.use(fileUpload({
2   createParentPath: true
3 })))
4 app.post('/api/concert/video', async (req, res) => {
5   console.log(req.files)
6 })

```

2.3 Connexion à un concert

La connexion à un concert se fait grâce à un cookie. Ce cookie est une chaîne de caractères générée à la création du concert par le serveur. Le serveur communique le cookie principal au créateur du concert. Celui-ci peut ensuite communiquer avec le serveur en fournissant ce cookie. La présence du cookie dans le message définit l'origine et la cible du message. Un autre cookie est donné aux autres utilisateurs se connectant simplement au concert sans l'avoir créé. Ce cookie permet de se connecter au concert via `Socket.io`, et uploader les vidéos sans son vers le serveur. Le cookie principal permet en plus de lancer et d'arrêter le concert. De plus la vidéo envoyée grâce au cookie principal sera celle dont l'audio sera utilisé pour la vidéo fusionnée.

3 Le serveur Socket.IO

Pour synchroniser l'envoi régulier des vidéos, le serveur doit pouvoir communiquer avec les téléphones, et faire le chef d'orchestre.

Pour cela on utilise les websockets. Cette technologie permet de créer une connexion bidirectionnelle entre le serveur. Une fois la connexion établie, le serveur peut envoyer des messages au client et vice-versa.

3.1 Exemple

`Socket.io` est une librairie permettant de communiquer en utilisant les websockets. Comme avec les url, `Socket.io` permet de définir des messages qui peuvent être reçus pas le serveur et exécuter une fonction à leur réception.

```

1 import http from 'http';
2 import express from 'express';
3 import { Server } from 'socket.io';
4 const server = http.createServer(app);
5 //Lancement du serveur
6 server.listen(3000,
7   () => console.log('App listening on port ${port}!'));
8 const io = new Server(server);
9
10 //Ecoute d'une connexion d'un utilisateur
11 io.on('connection', (socket) => {
12   console.log('utilisateur connecté');
13
14   socket.on('rejoindre_concert', (data) => {

```

```

15      //Le message rejoindre_concert a été reçu
16      socket.emit('OK');
17  });
18
19  });

```

L'objet socket décrit la connexion avec l'utilisateur. Cet objet peut donc être stocké pour communiquer dans l'autre sens.

3.2 Messages

Les messages utilisés sont :

- Messages reçus par le serveur
 - rejoindre_concert : Permet de rejoindre un concert avec son cookie
 - demarrer_concert : Permet de démarrer un concert avec son cookie
- Messages envoyés par le serveur
 - concert_rejoint : Validation du joignage d'un concert
 - demarrer_concert : Signale que le concert a commencé
 - checkpoint_video : Signale qu'il faut enregistrer la vidéo et l'envoyer
 - fin_concert : Signale la fin du concert

Tous les utilisateurs qui se seront connectés au concert en feront parti. Il est quand même possible d'en rejoindre un en route. Le cookie envoyé définit le concert que l'utilisateur veut rejoindre.

Une fois le concert démarré, le serveur va envoyer un message "demarrer_concert" à tous les utilisateurs connectés. Il faut alors commencer l'enregistrement d'une vidéo.

Puis toutes les 5 min le serveur va envoyer un message "checkpoint_video", cela signifie qu'il faut enregistrer la vidéo et l'envoyer au serveur.

Si aucun utilisateur n'est connecté au concert via websocket, le concert est supprimé au bout d'une minute.

4 Fusion des vidéos

Une fois toutes les vidéos envoyées lors d'un checkpoint, il faut les mélanger en conservant le son de la vidéo principale. Chaque vidéo est montrée, dans l'exemple où il y a 3 vidéos, 1/3 du temps. Les vidéos tournent en round-robin, dans l'ordre de leur apparition dans la liste donnée. Donc en l'occurrence, leur ordre d'arrivée vers le serveur. On peut retrouver cette opération dans le fichier videoEditing.js

On peut noter que si une seule vidéo a été envoyée, aucune opération n'est à réaliser.

4.1 FFMPEG

FFMPEG est un outil d'édition de fichiers vidéos et audios en ligne de commande. Il permet de découper, encoder, fusionner, etc.

On peut donc l'utiliser avec la méthode spawn du module natif Node.js qu'est `child_process`.

Avec plusieurs vidéos il faut séparer les opérations de découpage, assemblage et transfert de l'audio.

4.2 Découpage des vidéos

Avec FFMPEG il est possible de récupérer partie d'une vidéo

```
1 ffmpeg -ss 5 -i inputVideo.mp4 -t 10 \
2 -c:v copy -c:a copy \
3 outputVideo.mp4 \
```

L'argument -ss permet de définir un point de départ pour le flux d'entrées dans la video d'input et -t permet de définir la durée de récupération. Par exemple, pour l'exemple ci-dessus, on récupère 10 secondes de la vidéo à partir de la 5ème seconde. L'option -c permet de définir l'encodage, ici on conserve l'encodage originel de la vidéo. On peut répéter cette opération sur chaque vidéo en sauvegardant les chemins de sortie de chaque vidéo.

4.3 Concaténation des vidéos

La concaténation est un petit peu différente, pour dire à FFMPEG quelles vidéos concaténer, il faut lui donner un fichier contenant la liste des vidéos à récupérer.

Il faut donc écrire ce fichier et lancer ffmpeg dessus ensuite avec l'option "-f concat".

```
1 for(let path of shorterVideosPaths){
2   videoList += "file " + path + "\n";
3 }
4 try{
5
6   //Concaténation de toutes les vidéos
7   promises.writeFile("fichiersAConcatener.txt", videoList, 'utf8',
8   (err) => {
9     //Ecriture du fichier terminée
10    if(err) throw err
11
12    let concatProcess = spawn(pathToFFMPEG,
13    [ '-y', '-f', 'concat', "-safe", '0',
14      '-i', 'fichiersAConcatener.txt',
15      '-c', 'copy', shorterVideosPaths[0] ] );
16    });
17 }catch(err){
18   console.log(err)
19 }
```

Avec ce code on se retrouve donc avec la vidéo concaténée à l'emplacement de la première vidéo. Il ne reste plus que mettre le son de la vidéo principale sur celle-ci.

4.4 Transfert de l'audio

FFMPEG nous permet de choisir la modalité audio/vidéo selon un des fichiers d'entrée, grâce à l'argument map.

```
1 spawn(pathToFFMPEG,
2 [ '-y',
3   '-i', shorterVideosPaths[0],
```

```

4  '-i', audioPath,
5  '-c:v', 'copy', '-c:a', 'copy',
6  '-map', '0:v:0' //v=vidéo De l'entrée 0, vers la sortie 0
7  '-map', '1:a:0' //a=audio De l'entrée 1, vers la sortie 0
8  outputFile
9  ]);

```

On a donc bien finalement, une vidéo de sortie composée à 1/n de chaque vidéo avec par dessus le son de la vidéo principale.

Maintenant que la vidéo est prête, il faut la poster sur Facebook.

5 Diffusion sur Facebook

L'accès à l'API Facebook m'ayant été refusé, il faut utiliser un navigateur et simuler le fait de poster la vidéo manuellement.

Pour cela il faut utiliser **Puppeteer**.

Vous pouvez retrouver la fonction réalisée ici dans le fichier FacebookPoster.js.

On peut grâce à Puppeteer, naviguer dans un navigateur sans interface. Par exemple, dans le code suivant, on peut ouvrir une page dans une résolution donnée.

```

1  import puppeteer from 'puppeteer';
2  //Lancement du navigateur headless
3  const browser = await puppeteer.launch({ headless: true });
4  const page = await browser.newPage();
5  await page.setViewport({
6  width: 1920,
7  height: 1080,
8  deviceScaleFactor: 1,
9  })

```

Les fonctions qui nous sont le plus utiles sont “goto” pour se rendre à un certain URL et “click”, qui permet de cliquer à des certaines coordonnées sur la page actuellement ouverte. (Il existe un moyen de cibler certaines balises HTML pour cliquer directement dessus sans avoir à connaître l'emplacement, mais le site de Facebook est fait de telle façon qu'il est impossible de cibler une balise précise, toutes portent le même nom.) On peut se servir des ces 2 fonctions pour se connecter au compte.

Il est possible de prendre un screenshot de l'état actuel de la page naviguée, utile pour nous, pour déboguer où le système en est.

On peut ainsi en définissant des coordonnées et des temporisations, naviguer jusqu'à la page où poster la vidéo. On peut cliquer sur le bouton pour poster une vidéo, mais il faut ensuite sélectionner un fichier. Pour cela, il faut utiliser un objet FileChooser, qui prend en argument une liste de fichier à entrer dans la fenêtre de sélection.

```

1  //SELECTION DU FICHIER A UPLOADER
2  const [fileChooser] = await Promise.all([
3    page.waitForFileChooser(),
4    page.mouse.click(1200,485) //CLICK SUR LE BOUTON PHOTO/VIDEO
5  ]);
6  await fileChooser.accept([cheminVideo]);
7  //Attente de l'upload du fichier

```



Figure 5.2 – Vue juste avant la connexion

```
8 | await new Promise(d => setTimeout(d, 20000));
```

On peut ensuite attendre et cliquer sur le bouton publier.

Cette méthode utilise beaucoup de temporisations, normalement avec un temps assez généreux, mais est donc assez lente, prône aux erreurs en cas de latence du réseau et peut casser facilement en cas de modification de l'interface par Facebook.

6 Application mobile

6.1 Introduction

Pour s'interfacer avec le serveur, il faut l'application mobile. L'application doit être capable de créer/rejoindre un concert et filmer et uploader des vidéos pour y participer.

L'application est réalisée avec le framework Flutter basé sur le langage Dart. Ce langage permet de développer des applications pour Android/iOS mais aussi depuis plus récemment, application desktop. On peut retrouver la documentation du framework ici : <https://docs.flutter.dev/get-started/install>

6.2 Séparation des éléments

Le découpage de l'application est plutôt simple. Il suffit de regarder l'interface. Chaque élément de l'interface possède son fichier. Chaque élément cliquable possède une fonction click qui s'exécute lorsque l'on clique sur le bouton. Et chaque élément possède une fonction initState, qui s'exécute au rendu du widget.

Pour vision plus précise que les explications présentées ici, se référer à la dartDoc -> `"/doc/api/index.html"`

```
1 | camera.globals //Fichier contenant les variables globales
2 | concert //Modèle de données d'un concert en local
3 | concertPage //Page de la vue connectée à un concert
4 | creerConcertButton //Bouton pour créer un concert
```

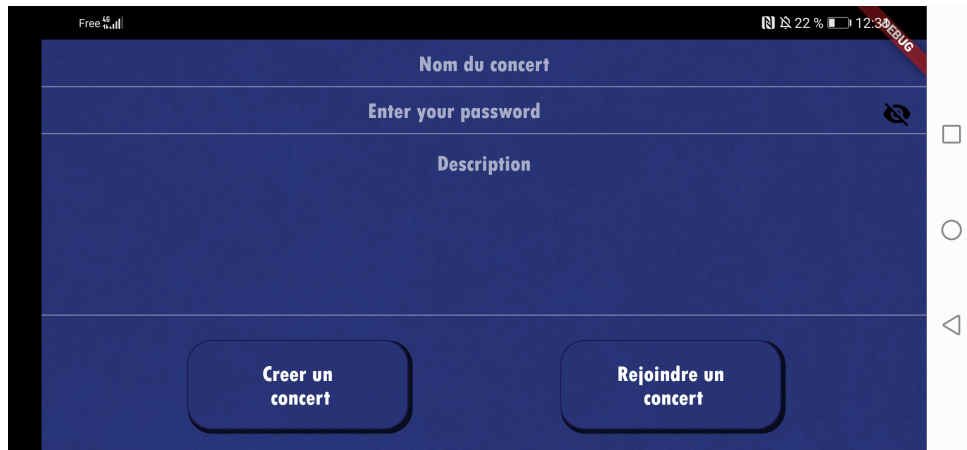


Figure 5.3 – Vue menu principal

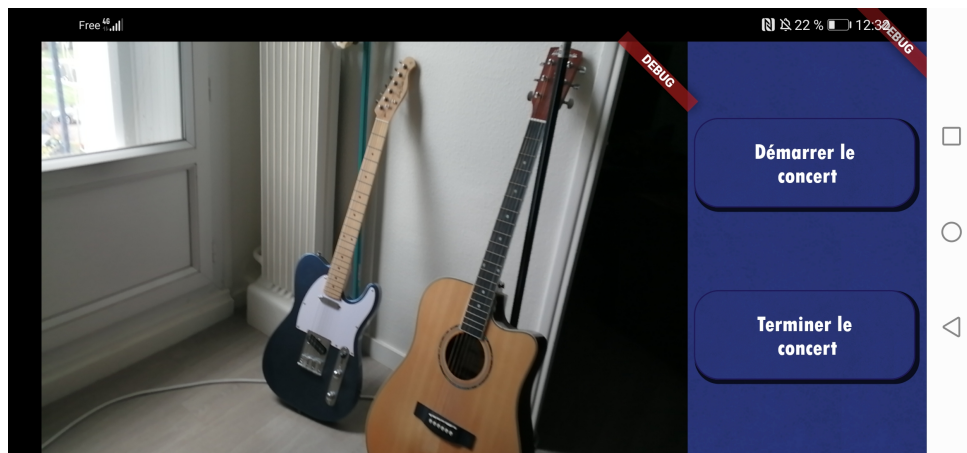


Figure 5.4 – Vue durant un concert

```

5 | demarrerConcertButton //Bouton pour démarrer un concert
6 | main //Point d'entrée du programme
7 | mainMenu //Menu principal
8 | previewCamera //Preview de la caméra dans la vue concert
9 | rejoindreConcertButton //Bouton pour rejoindre un concert
10 | routes //Objet décrivant les liens entre les pages
11 | terminerConcertButton //Bouton pour

```

Le coeur de la logique du programme se trouve dans les fonctions click des différents boutons. Aucune action n'est effectuée tant que l'utilisateur n'appui pas sur un bouton.

6.3 Librairie de communication

La librairie de communication (se trouvant dans /communication), permet d'accéder aux différentes URLs de l'API Inside. Elle permet de relier de communiquer les données ainsi qu'exécuter des callbacks lors de réceptions de messages.

Pour toutes les informations sur ce module, lire sa dartDoc "<communication/doc/api/index.html>"

6.4 Caméra et variables globales

L'utilisation de la caméra utilise la librairie "camera" de flutter.

Tout d'abord il faut initialiser un contrôleur de caméra qui permettra d'interagir avec.

```

1 WidgetsFlutterBinding.ensureInitialized();
2 //Recherche des caméras disponible
3 availableCameras().then((value) => globals.cameras = value);
4 //....
5 //initialisation du contrôleur sur la caméra 0
6 globals.cameraController = CameraController(
7     globals.cameras[0], ResolutionPreset.max,
8     enableAudio: true);
9
10 //Lancement d'un enregistrement
11 globals.cameraController.startVideoRecording();
12 //Arrêt d'un enregistrement et envoi de la vidéo
13 globals.cameraController?.stopVideoRecording()?.then((file) {
14     print('envoi de la vidéo: ' + file.path);
15     Communication.uploadVideo(file.path);
16     globals.cameraController.startVideoRecording();
17 });

```

7 Bugs et améliorations futures

Durant la réalisation de l'application j'ai tout de même fait face à certains problèmes. Il reste 3 problèmes principaux dans le fonctionnement de l'application :

7.1 Vidéo renversée

Pour une raison que j'ignore, la vidéo enregistrée par l'application mobile est à l'envers. On peut résoudre ce problème, simplement en retournant le téléphone en mode rotation automatique. L'interface restera à l'endroit et la vidéo se remettra à l'endroit par retournement du téléphone. Mais, ce fix n'est que temporaire, puisque lors d'un checkpoint, la vidéo va repartir et le problème de retournement avec. La preview de la caméra, elle, reste à l'endroit.

7.2 Chevauchement des checkpoints

Je n'ai pas défini d'identifiant pour chaque checkpoint lors d'un concert. Ce qui peut créer des soucis si la vidéo d'un checkpoint arrive alors que la vidéo du checkpoint précédent n'a pas fini d'être traitée.

Il faudrait un identifiant par checkpoint pour vérifier qu'une vidéo arrivée concerne bien ce checkpoint.

Mais cette méthode vient également avec son lot de problème, que se passe-t-il si aucune vidéo n'arrive ? Si une vidéo arrive mais après le début d'un autre checkpoint ?

Des questions auxquelles je n'ai pas eu le temps d'apporter une réponse.

7.3 Facebook

Il est impossible de modifier un post Facebook, alors, le module pour interagir avec Facebook ne permet que de poster une vidéo. Si l'on souhaite supprimer le post précédent pour juste avoir le

plus direct, cela créerai un décalage, à cause du temps que facebook met à traiter le post d'une vidéo comparé à l'instantanéité de la suppression.

Il faudrait dans le meilleur des cas, obtenir l'autorisation de l'API par Facebook. Sinon, laisser ainsi et n'avoir que des posts non ordonnés de concerts potentiellement différents.

6

Bilan et conclusion

1 Bilan du semestre 9

J'ai bien dépoussiéré le projet, en terme de fonctionnalités à réaliser et en termes de faisabilité. Les différents tests et recherches que j'ai fait me confortent dans la réalisation concrète du projet. Les tâches pour le prochain semestre ont été bien définies et ne devrait à priori pas causer de retard.

2 Bilan du semestre 10

La réalisation a été retardée par le refus de Facebook pour l'utilisation de l'API. Mais j'ai tout de même pu terminer à temps un rendu fonctionnel.

J'ai pu développer mes compétences en développement d'API, car c'est quelque chose que je n'avais jamais vraiment fait. Ce projet était assez complet en terme d'interface, de back et d'utilisation de librairies externes.

Je suis content d'avoir pu arriver à mes fins. Malgré les soucis, que j'ai évoqué lors de la section précédente, il y a tout de même un rendu fonctionnel.

Annexes

A

Planification, gestion de projet

1 Evolution du projet

Le diagramme de Gantt Initial pour la planification de ce projet

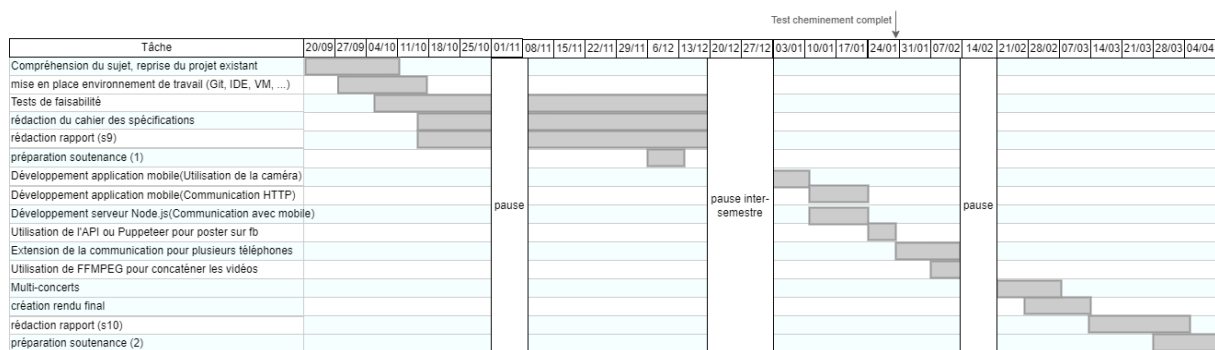


Figure A.1 – Le diagramme de Gantt Initial

Le diagramme de Gantt Final pour la planification de ce projet

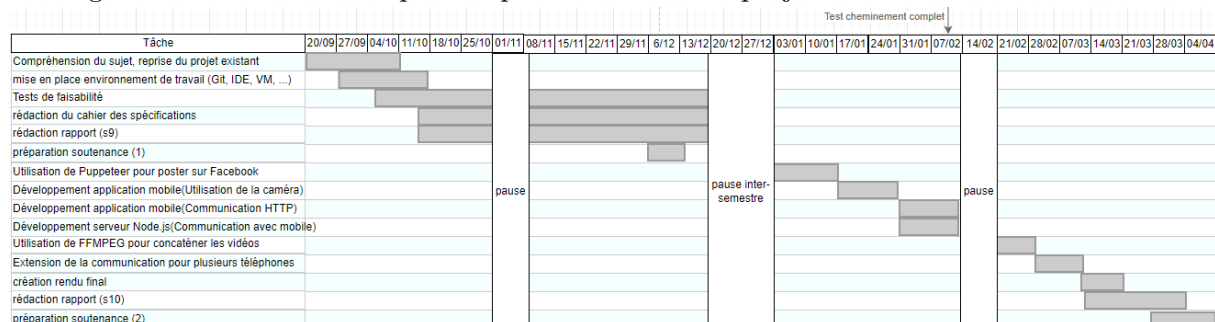


Figure A.2 – Le diagramme de Gantt Final

B

Cahier de Spécifications

1 Contexte de la réalisation

1.1 Objectifs

En temps de pandémie, il est difficile d'organiser des concerts, et encore moins avec beaucoup de monde. Sylvain Fabre, musicien professionnel, souhaiterait donc un moyen facile de diffuser les concerts sur internet en direct pour faire profiter un maximum de personnes même à distance.

Le projet Inside cherche à résoudre ce problème en permettant aux musiciens de diffuser leurs concerts sur une plate-forme en ligne.

De plus, Inside propose différents points de vue, à l'aide de caméras portées directement par les musiciens pour des points de vue inédits.

L'objectif du projet est de mettre en place ce système de diffusion audio/vidéo sur des réseaux sociaux, d'une façon accessible aux musiciens ainsi qu'aux spectateurs.

1.2 Hypothèses

Ce qui est susceptible de changer dans le projet est l'utilisation de l'API Facebook. Il faut une approbation de l'application par Facebook, si cette approbation est trop difficile à obtenir, il faudra se contenter d'une solution de simulation de navigateur web pour poster les vidéos « manuellement » sur Facebook.

1.3 Bases méthodologiques

Pour mettre à bien ce projet, j'ai mis en place un planning prévisionnel contenant les grandes tâches à réaliser.

J'ai également mis en place un Git pour, dans un premier temps, pouvoir conserver mon travail entre plusieurs machines, et faire du versioning pour le développement du système.

2 Description générale

2.1 Environnement du projet

Inside est une plate-forme de diffusion de concerts en direct. Plusieurs caméras et un micro sont disposés sur scène, les flux audio et vidéo sont récupérés puis envoyés sur une plate-forme en ligne. L'installation des caméras et micro est faite par le/les artistes présents.

Ce système a déjà été réalisé lors de précédents projets. Des caméras spéciales et un PC devaient être présents sur scène. Le PC récupérait les flux des caméras et les retransmettaient sur un site web en direct. Le site web permettait de changer de vue entre les différentes caméras grâce à une mosaïque.

L'objectif principal du projet est d'adapter ce système en se passant à la fois du PC et des caméras sur scène et du site web.

2.2 Caractéristiques des utilisateurs

Les premiers utilisateurs du système sont des musiciens non spécialistes techniques. Le système doit donc être facile d'utilisation et transparent pour l'utilisateur. Les musiciens doivent pouvoir utiliser sans connaissances informatiques poussées.

De l'autre côté du système, les utilisateurs finaux sont les spectateurs en ligne, ils doivent pouvoir accéder à la diffusion du concert en direct facilement sur une page d'un réseau social.

2.3 Fonctionnalités du système

Le système doit être capable de :

- Créer un concert en ligne
- Assigner des téléphones à un concert
- Diffuser un flux vidéo composé des différentes vues des téléphones sur un réseau social

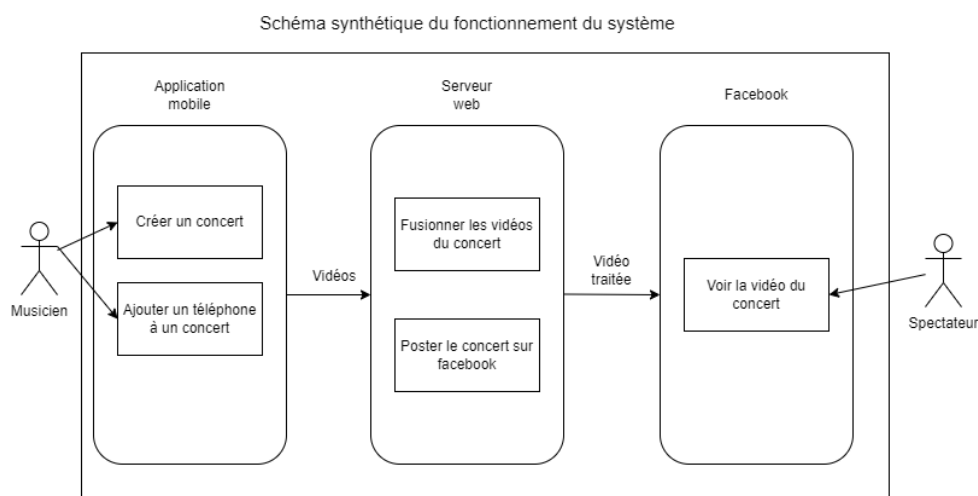


Figure B.1 – Schéma synthétique du système

3 Description des interfaces externes du logiciel

3.1 Interfaces matériel/logiciel

L'application mobile doit pouvoir être utilisée par le plus grand nombre et donc être disponible sous Android et IOS. Le déploiement d'une application sous IOS étant complexe, il ne sera probablement pas possible de le réaliser mais néanmoins, la technologie utilisée pour réaliser l'application doit permettre un déploiement multi-plateforme.

Le serveur sera hébergé par un service en ligne à déterminer, probablement une machine linux. Dans le doute, le serveur sera réalisé avec une technologie déployable sous Linux et Windows.

3.2 Interfaces homme/machine

La navigation dans l'application se fera par des musiciens non spécialistes informatiques, l'interface doit donc être simple d'utilisation.

Ci-après, des schémas des vues des différentes pages de l'application.

Il y aura sur l'application 2 pages :

- Une page d'accueil de l'application permettant de créer un concert ou en rejoindre un
- Une page avec une vue caméra pour visualiser ce qui est envoyé au serveur pour le concert avec des boutons pour démarrer ou arrêter le concert

Vue page d'accueil

Le diagramme illustre la structure de la page d'accueil. Il est composé d'un conteneur principal rectangulaire qui abrite quatre éléments alignés verticalement : un champ de texte pour le 'Nom du concert', un champ de texte pour le 'Mot de passe', un grand champ de texte pour la 'Description', et deux boutons rectangulaires à coins arrondis. Le premier bouton est étiqueté 'Créer un concert' et le second 'Rejoindre un concert'.

Figure B.2 – Page d'accueil de l'application

Vue téléphone principal

Figure B.3 – Vue horizontale, concert créé sur le téléphone principal

Vue téléphone principal

Figure B.4 – Vue verticale, concert créé sur le téléphone principal

Un concert est défini par un nom et un mot de passe, et éventuellement une description. Après avoir défini ces paramètres, on peut donc créer un concert ou rejoindre un concert existant. La vue passe donc sur la vue concert en cours. Un téléphone qui rejoint un concert mais ne le crée pas n'aura pas accès aux boutons de démarrage/arrêt.

3.3 Interfaces logiciel/logiciel

Entre les différentes parties du système, la communication entre l'application mobile et le serveur web se fera via la connexion 4G/5G du téléphone et le protocole http. La connexion entre le serveur et Facebook se fera également grâce au protocole http.

4 Spécifications fonctionnelles

Le système doit être capable de :

- Créer un concert en ligne

Sur l'application le musicien utilisateur peut créer son concert qui sera rediffusé sur la page Facebook Inside. Après avoir entré le nom et le mot de passe et la description du concert le téléphone utilisé sera le téléphone principal du concert. Ce téléphone sera celui sur lequel le son sera récupéré. Le son des autres téléphones est ignoré. Ceci est pour maximiser la qualité sonore. Le concert est existant côté serveur et d'autres téléphones peuvent s'y connecter mais il n'est pas encore commencé, aucune donnée vidéo/audio n'est envoyée, le concert n'est pas encore sur Facebook. Le téléphone passe ensuite sur une vue de la caméra pour une prévision du flux envoyé.

- Assigner des téléphones à un concert

Une fois le concert créé via le téléphone principal, il est possible d'ajouter d'autres téléphones au concert. Ces autres téléphones définissent des nouvelles prises de vue qui seront intégrées dans le flux diffusé sur Facebook. Pour ajouter un téléphone à un concert il faut en renseigner le nom ainsi que le mot de passe.

- Démarrer le concert

Une fois le concert créé, le téléphone principal peut démarrer le concert. A ce moment, les téléphones commencent à enregistrer des vidéos via la caméra et envoyer ces vidéos vers le serveur. Les vidéos feront une durée encore indéterminée, qui dépendra des tests effectués.

- Diffuser un flux vidéo composé des différentes vues des téléphones sur un réseau social

Après l'envoi des premières vidéos par les téléphones, le serveur va les traiter pour en faire une seule vidéo, en montrant les différentes vues les unes après les autres. La vidéo ainsi créée est ensuite postée sur la page Facebook avec la description entrée par l'utilisateur. Cette vidéo est mise à jour à chaque nouvelle vidéo créée par le serveur, permettant au spectateur d'être au plus proche possible du direct. La décision de conserver la vidéo entière du concert ou simplement les dernières minutes dépendra de la capacité du serveur à gérer cette vidéo et de la praticité du système pour le spectateur à l'usage.

Ci-après, diagramme de séquence reprenant le fonctionnement temporel du système.

5 Spécifications non fonctionnelles

5.1 Contraintes de développement et conception

Il faudra pour développer ce système, au moins 2 téléphones Android prêtés par le service informatique de Polytech. Le serveur pourra être un ordinateur quelconque pour les tests. Pour le déploiement réel du système il faudra un serveur loué par le client, à déterminer selon les tests futurs.

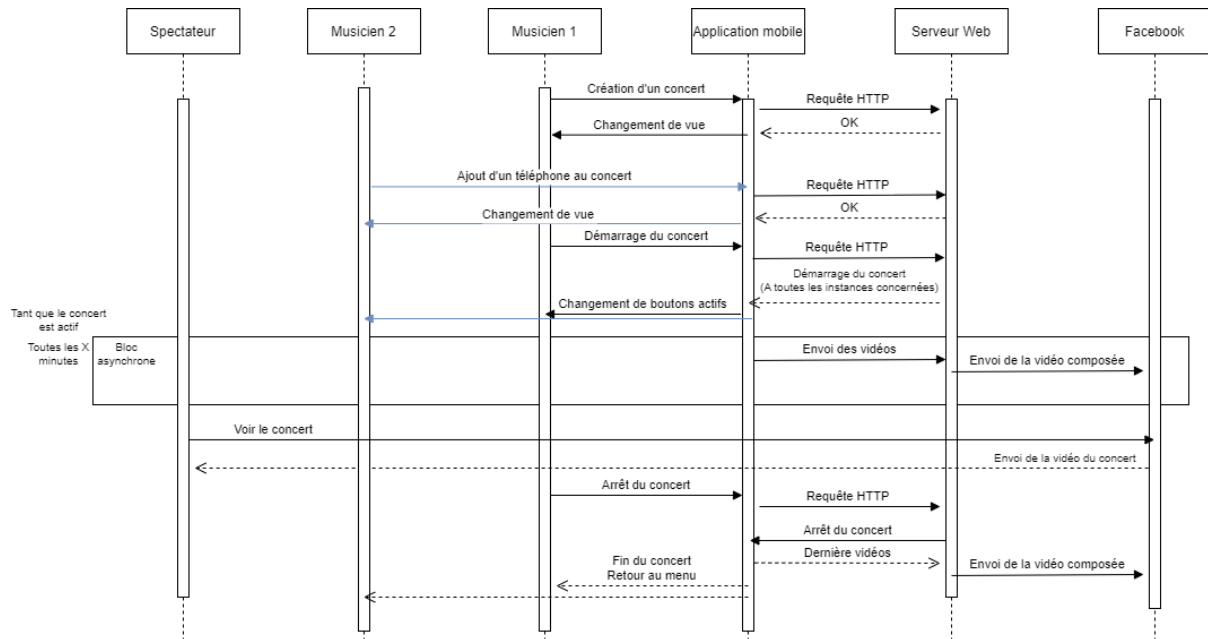


Figure B.5 – Diagramme de séquence du système

Les technologies utilisées : Pour l'application mobile, le langage Dart associé au framework Flutter. Cette technologie permet de créer des applications mobiles déployables sous IOS et Android. Toutes les fonctionnalités de communications réseau ainsi que de gestion de la caméra sont disponibles.

Pour le serveur web, la technologie utilisée sera Node.js. Node permet de créer des serveurs web à l'aide du langage javascript, c'est adapté à de petits nombres de connexions simultanées. La communication entre les téléphones et le serveur se fera probablement via 3G/4G/5G, utilisant possiblement les websockets.

FFMPEG sera utilisé pour le traitement des vidéos reçues par le serveur. FFMPEG est un outil logiciel de traitement vidéo/audio. Node.js permet l'utilisation de FFMPEG.

La communication avec Facebook se fera d'une de deux façons selon l'acceptation de l'application par Facebook. L'utilisation de l'API Facebook nécessite une approbation de l'application de la part de Facebook. Si cette condition est satisfaite, l'API Facebook sera utilisée pour poster les vidéos sur la page.

Si l'approbation est refusée ou met trop de temps à être acceptée, il faudra utiliser un outil de simulation de page de web, par exemple Puppeteer.

5.2 Capacité

Il a été convenu que pour l'instant le système devra supporter 1 seul concert. Cependant le système doit être pensé pour être agrandissable lors de sa conception.



Document d'installation

1 Serveur

1.1 Node.js

L'application est basée sur la technologie Node.js et le framework Express Il faut donc tout d'abord télécharger Node.js [[https ://nodejs.org/en/download/](https://nodejs.org/en/download/)]([https ://nodejs.org/en/download/](https://nodejs.org/en/download/))

Il faut ensuite installer les paquets nécessaires au fonctionnement du programme, pour cela, ouvrir un terminal à la racine du projet et exécuter la commande “npm i”

1.2 FFMPEG

FFMPEG Permet le mélange des vidéos.

Il faut donc le télécharger <https://www.ffmpeg.org/download.html> Et donner le chemin relatif depuis la racine du projet vers l'exécutable ffmpeg dans le fichier “videoEditing.js”

1.3 Page facebook

Pour poster les vidéos sur facebook, il faut créer un compte ainsi qu'une page.

Pour pouvoir s'y connecter il faut renseigner l'email et le mot de passe dans des fichiers nommés “email” et “password” à la racine du projet.

Il faut également renseigner l'URL vers la page où poster les vidéos dans la variable “pageURL” en haut du fichier “index.js”

2 Application mobile

L'application est basée sur la technologie dart/flutter. Pour installer l'environnement il faut suivre le tutoriel sur le site de flutter : <https://docs.flutter.dev/get-started/install>

Il faut ensuite exécuter la commande “flutter pub get” à la racine du projet pour télécharger toutes les dépendances.

D

Document d'utilisation

1 API

c.f. fichier “API_README.md”

2 Application Mobile

L'application permet de retransmettre le flux vidéo de la caméra en direct sur le page Facebook de l'application Inside.

A l'ouverture de l'application, des champs sont à remplir pour fournir les informations du concert. On peut ensuite soit créer un concert ou rejoindre un concert existant.

Si le concert a bien pu être créé/rejoint, on passe à la vue concert. Il y a une preview de la caméra et la possibilité de démarrer le concert. Les vidéos des différents point de vue des téléphones connectés au concert sont ensuite mélangées et diffusées sur Facebook.

Le concert peut être démarré et arrêté par la personne l'ayant créé.

Solution de capture et composition temps-réel de flux video sur mobile : Inside

Adrien PREVOST

Encadrement : Mathieu DELALANDRE



En collaboration avec Polytech

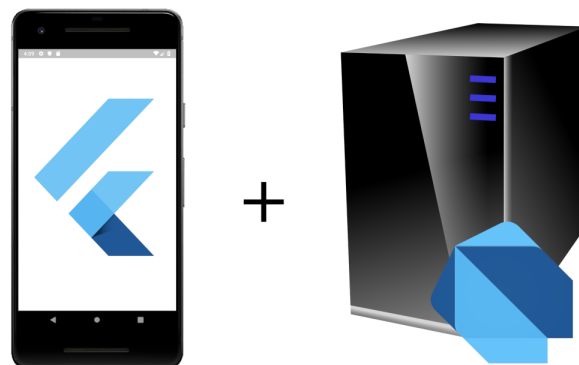
Objectifs

- Permettre à des musiciens de retransmettre facilement un concert sur les réseaux sociaux
- Permettre des prises de vue multiples
- Rendre l'accès à ces fonctionnalités le plus facile possibles



Mise en œuvre

- Recherche d'une architecture client/serveur capable d'être interfacé sur mobile
- Création d'un système de mixage d'audiovisuel
- Développement d'un module d'interaction avec Facebook



Résultats attendus

- Application mobile permettant de créer un concert et de le filmer depuis différents téléphones
- Application serveur capable de retransmettre le concert sur Facebook



Solution de capture et composition temps-réel de flux video sur mobile : Inside

Adrien PREVOST

Encadrement : Mathieu DELALANDRE

Objectifs

- Permettre à des musiciens de retransmettre facilement un concert sur les réseaux sociaux
- Permettre des prises de vue multiples
- Rendre l'accès à ces fonctionnalités le plus facile possibles

Mise en œuvre

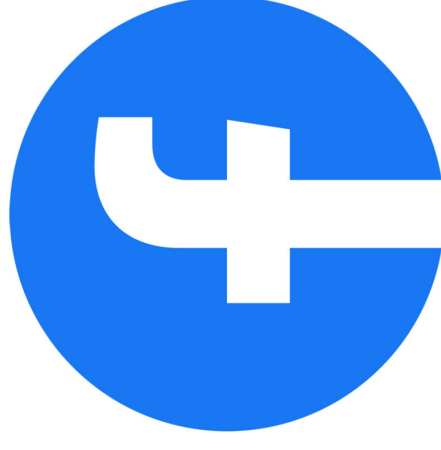
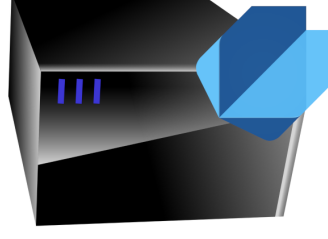
- Recherche d'une architecture client/-serveur capable d'être interfacé sur mobile
- Création d'un système de mixage d'audiovisuel
- Développement d'un module d'interaction avec Facebook

Résultats attendus

- Application mobile permettant de créer un concert et de le filmer depuis différents téléphones
- Application serveur capable de retransmettre le concert sur Facebook



+



Solution de capture et composition temps-réel de flux video sur mobile

Inside

Résumé

Ce projet consiste à porter sur plateforme mobile un système de retransmission de concert, sous différents points de vue, sur internet en direct. Les musiciens doivent pouvoir utiliser leur téléphone pour filmer et enregistrer un concert que les spectateurs pourront regarder en direct sur Facebook. Le système existe déjà mais nécessite la présence de caméras particulières, d'un ordinateur directement sur scène et le rendu était diffusé sur un site web indépendant. Dans ce projet le pc local est remplacé par un serveur distant, les caméras sont remplacées par les téléphones des musiciens et le site web est remplacé par une page Facebook. Ce projet contient donc du développement mobile, du développement web ainsi que du traitement de flux vidéo.

Mots-clés

Développement mobile, Flux vidéos, Node.js, FFmpeg, API

Abstract

The goal of the project is to update a concert live broadcasting system. The system allows musicians to broadcast their concerts on an website, from different points of view. The system already exists but requires specific cameras and a computer on stage, as well as a dedicated website. The goal is to replace all those things by simpler ones. The specific cameras are replaced by smartphones, and the local pc disappears. The livestream is also replaced by a social network, in our case Facebook. The project requires mobile development, web development as well as video processing and Facebook API use. The application will be made using Flutter, and the web server with Node.js.

Keywords

Flutter, Mobile development, Web development, Video processing, Node.js, Flutter, FFMPEG

Entreprise

Polytech



Tuteur entreprise

Sylvain FABRE

Étudiant

Adrien PREVOST (DI5)

Tuteur académique

Mathieu DELALANDRE