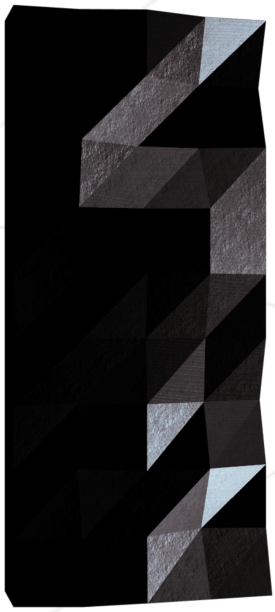

Music Room

Music, Collaboration and mobility

Summary: Creation of a complete mobile solution focused on music and user experience.

Contents

Chapter I

Preamble

I.1 The musical experience

The musical "consumption" is a personal experience everyone lives in their own personal way:

- Surrounding with music everyday,
- Enjoying music when sharing time with friends,
- Using music to isolate oneself or focus,
- Only believing in performance, live shows or festivals.

Chapter II

Objectives

This project aims to tackle all the concepts necessary to the creation of a mobile, connected and collaborative application taking into account the constraints of a real product.

You will have to work on the client/server architecture of you solution, take decisions for you data storage, define an API that will be used as a communication channel between your server and your clients, anticipate the problems of ramp-up and securing, and learn to work with third parties, integrating external SDKs.

There are a lot of subjects. You should dispatch the different tasks between your project team members.

The following services will have to be implemented:

- **Music Track Vote** - Live music chain with vote,
- **Music Control Delegation** - Music control delegation,
- **Music Playlist Editor** - Real time multi-user playlist edition

`Warning the SDK you choose must not do your work.`

Chapter III

General instructions

III.1 Instructions concerning the software architecture

There is no constraint as to strategies or language you must use for the back-end: you will have to evaluate the benefits and disadvantages of the different available technologies and identify the ones that will suit the project best. You will have to justify your decisions.

You cannot commit libraries you would not have written yourself in your repo. Your project's dependancies must be automatically downloadable from a clone of your repo thanks to a Makefile or similar mechanisms.

III.2 Instructions concerning the mobile experience

The mobile application must be implemented either for **Android** (Java, Kotlin, even NDK or Xamarin) or for **iOS** (Objective-C, Swift or Xamarin). The user will be able to perform any action from the mobile application.

Chapter IV

Mandatory part

IV.1 User

When the user runs the application for the first time, they must create an account on the application. They will choose between a mail/password or a social network account (**Facebook** ou **Google**) registration.

Once the user has an account on your platform, the application must allow them to link their network account (**Facebook** ou **Google**).

In their profile, the user must be able to state and update:

- their public informations,
- informations only available to their friends,
- their private informations,
- their music preferences.

`If the user has created an account with a mail/password, the application must demand a mail validation and invites the user to change their password if they forgot it.`

IV.2 Services

On the application, user must access at least 2 functions out of the following 3 ones: **Music Track Vote**, **Music Playlist Editor** and **Music Control Delegation**.

IV.2.1 Music Track Vote

`Live music chain with vote.`

Some people are gathered in a place (party, event...). Your service must allow anyone to suggest or vote for the next track coming in the current playlist. If a track gets many votes, it goes up in the list and is played earlier.

Visibility management (Public/Private) must be integrated to the service:

- By default, your event is public.
- All the users can find your event and vote if your event is public.
- Invited users are the only ones who can find the event and vote if your event is private.

A license management must be integrated to the service:

- By default, everyone can vote.
- With the right license, the invited persons are the only one who can vote.
- With the right license, the persons located in a specific place for at a specific time (between 4 and 6PM for instance) will be able to vote.

`You should especially care about the management of competition problematics: for instance, if several persons vote for different tracks or the same one in a playlist.`

IV.2.2 Music Control Delegation

`Music control delegation.`

A license management must be integrated to the service. It must be specific for each device attached to the user's account. The user can choose to give the music control to different friends.

IV.2.3 Music Playlist Editor

`Real time multi-user playlist`

Collaborate with your friends or people with the same musical tastes to create playlists in real-time. This way, users can create original radio stations.

A visibility management (Public/Private) must be implemented to the service:

- By default, a playlist is public.
- If it is public, every user have access to the playlist.
- If it is private, only the invited users can have access to the playlist.

A license management must be implemented to the service:

- By default, everyone can edit the playlist.
- With the right license, the invited users are the only ones who can edit the playlist.

`You should especially care about the management of competition problematics: for instance, if several persons move different tracks or the same one in a playlist.`

IV.3 Server

All the service data will be stored on the back-end side. The back-end is the reference. He is the keeper and representative of the "truth". You can use the technology and frameworks your like (php, nodejs, golang, firebase, etc.).

IV.4 API

The API will be the access point to your back-end for all the applications. Some developers will lean on your API for various context integrations. Hence, the documentation regarding this API is essential and you are the first developers who will use it. This API reference documentation must introduce methods, inputs and outputs. It can be self-generated with Swagger, for instance.

You should create an API that endorses the principles REST but there are others out there (new trends spawn on a daily basis). Anyway, you will have to be able to justify your choice and explain its characteristics.

You should endorse JSON for your exchange format with the API but there are others out there. Anyway, you will have to be able to justify your choice and explain its characteristics.

IV.5 Mobile application

Applications must only be "remote control" to the back-end and the back-end's address must be configurable on the application for tests.

Authentication support through a social network like (**Facebook** or **Google**) must be implemented in the mobile application. Resources are available on: developers.facebook.com and developers.google.com.

IV.6 Securing

An authenticated user on your solution must have access to their data, but not to other users' data. You must plan malicious behaviors (bruteforce of your API, session theft, etc.) but your API is not expected to be unbreakable:

- you must implement mechanisms to protect your users,
- you must identify other hazards and explain the practicable protections.

Any action on the mobile application must generate logs on the back-end.

- Platform (Android, iOS, etc.),
- Device (iPhone 6G, iPad Air, Samsung Edge, etc.),
- Application Version.

IV.7 Ramp-up

You must be able to evaluate the load your API and your back-end can support, that is, justify and measure the number of users that can simultaneously use your 3 services. You can use AB (Apache Benchmark), Gatling, Siege, Tsung, JMeter for instance.

Don't forget to specify the servers characteristics (CPU, RAM, Cloud or Premise, etc.). The maximum number of users should be consistent with the platform choice. Dozens for a Raspberry, thousands for a low-end server.

IV.8 Agility, quality and continued integration

You have a lot of layers and functionalities to implement for this project, and you will have to work in a team. You should arrange yourself to show agility, to be able to question your own decisions, and to set specific one-off tests for each layer.

Chapter V

Bonus part

The bonuses proposed here match real business problematics you will face when developing a mobile application.

V.1 Multi-platform support

Once you have a server and a functional API, you can make your services available on various platforms. You should already support a mobile platform (Android or iOS). As a bonus, you could also make your service web "responsive" so it can adapt to any screen size.

According to your technological choices, web support can require that you rewrite your client code entirely or almost entirely.

V.2 Reflexion on the IoT

You can implement a mechanism such as beacons on your event. For instance, when people get near a public event registered on your service, they automatically receive informations about the event, how to access it, the kind of music etc.

V.3 Free subscription vs. Paid subscription

Nowadays, web solutions and mobile applications often offer the user to choose between a free limited subscription and a several unlimited paid ones.

This bonus will allow you to implement this kind of logic in your application. You will then have to allow users to switch between two offers you will have established beforehand. Some of your application's functionalities (**Music Playlist Editor** for instance) will only be available to users with a paid subscription.

V.4 Offline Mode

With this bonus, you will implement a mechanism that allows the users to enjoy the application offline, when, for instance, he will have no mobile service whatsoever. In that case, the experience and functions proposed by the application might be completely

different.

If the application is used offline, you should plan a synchronisation. Beware, though, sync mechanisms carry their lot of problems!:

- Managing conflicts and concurrency,
- Managing obsolete data on the mobile application.

Chapter VI

Turn-in and peer-evaluation

VI.1 Turn-in

As usual, turn in your work on your repo `GiT`. Only the work included on your repo will be reviewed during the evaluation.

VI.2 Evaluation

Assessor will positively welcome the quality of your reflexion and the originality of your work.