Report of the Soundboard project

Maria Guy De Fontgalland

I / Introduction

As part of the classes, I am taking, I had some on React Native: this Javascript library allows developers to make web applications and user interfaces from scratch. We have to mention it is different from React.
The last project we have to work on is the Soundboard project. This project aims to create a web application that allows the user to play with samples. There will be multiple features on it which are the following:

- Play a sound when the user clicks on a button
- View his library of samples
- Add new sounds or samples to this library thanks to 2 methods :
    - By recording with the user's phone
    - By adding new sounds from the Freesound API
- Editing a sound (cropping a sound)

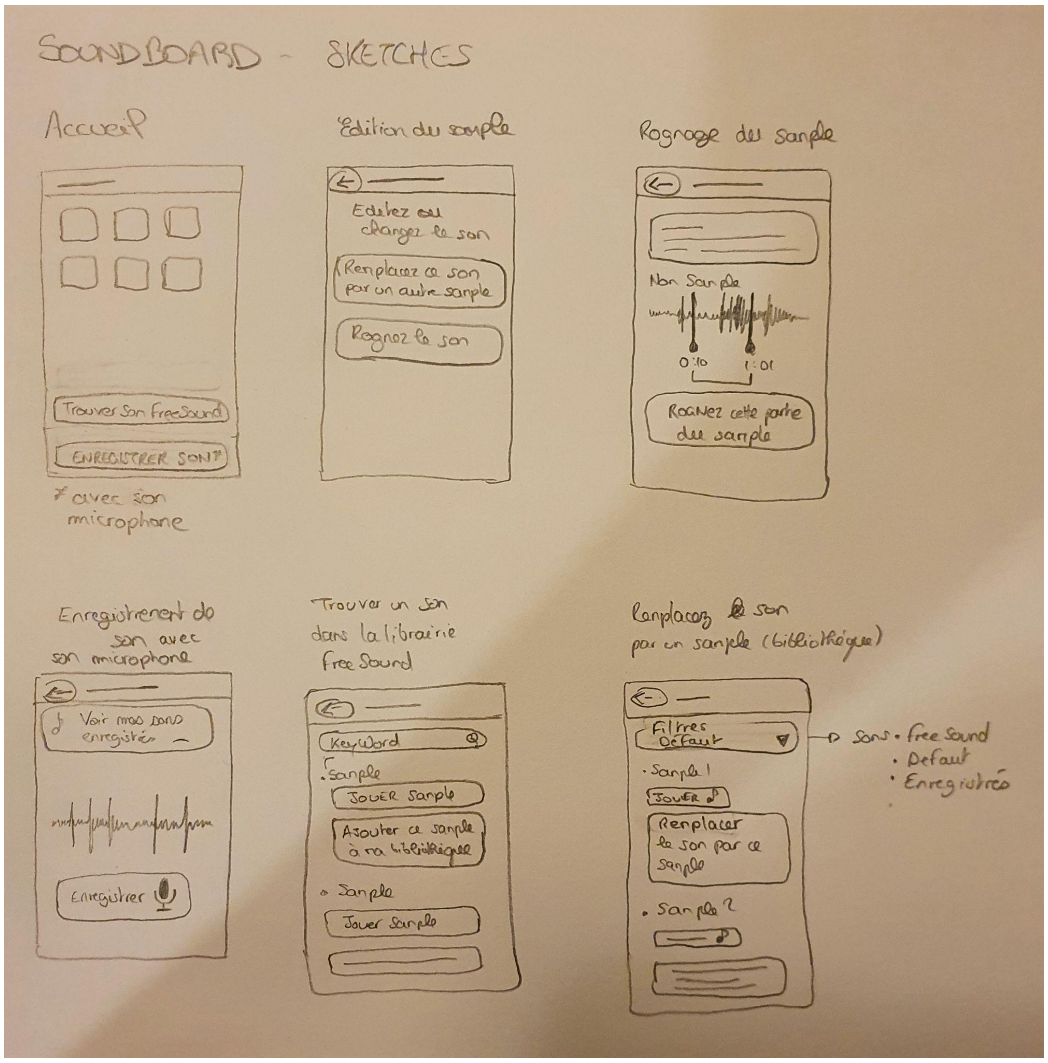
During the project, I could discover new tools that could help me develop my application. The first tool is the Google Chrome extension Redux DevTools (link to the Chrome Web Store): a friend introduced me to this tool and it was easier for me to work with Redux. I mainly used it to see the value of my state but it has other features that may be interesting.

Then we have the application ExpoGo. While running my code on the browser, I noticed in my terminal a message: it said that we can use this mobile application to test the code on a mobile device. It is so useful because you just have to scan the QR code in the terminal or on the first web page the localhost shows you and you can test it right away.

Then we have GitHub: we were asked to use this tool to host our code there and also manage the version control. You will find on my Github account the resources and the files for this project if you go to this link.

II / Structure of the application

In order to develop an application, it can be helpful to sketch the screens of the application: it will give an idea of what we need to take care of and how we are going to sort the data and the components.



From my sketches, I would have the following components :

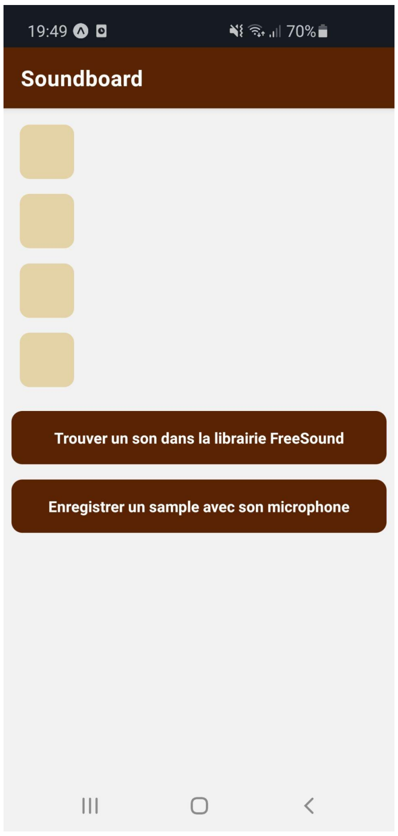
- The soundboard
- The menu that allows the user to choose to edit the sample
- The library of the sample
- The Freesound samples view with its search bar
- The Microphone view
- The view to crop/edit a sample

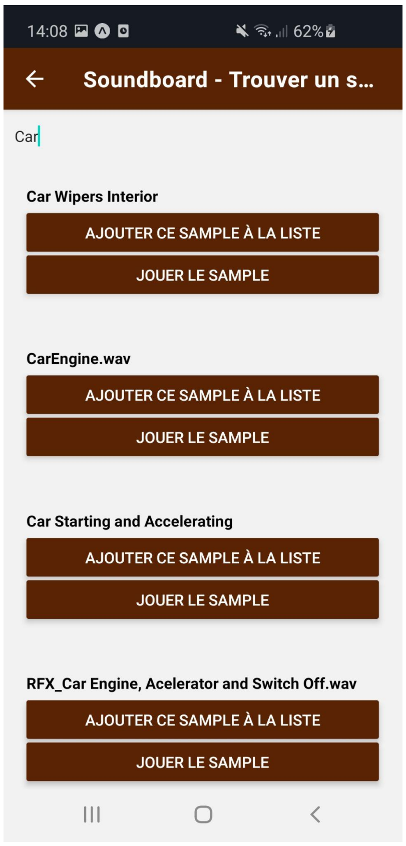
For this project, we are expected to use a Redux store: therefore we will stock the samples that the user uses for the soundboard in one slice and we will have another slice that has all the samples the user has in its library. And we will use it when it is necessary.
In the state, we will have a list of the samples. The data of one sample will be stocked in an object. Hence we will have a list of objects in the state.
In one object, we will find the following information:

- the name of the sample
- the id of the sample
- the URL of the sample
- the category: whether it is a default sample (default), a sample downloaded from the API (freesound), a sound that was recorded (recording), or an edited sample (edited)

III / Storyboard

The main scenario that the user can do with the current features I developed would be this one: the user plays with the soundboard (screen 1) but he wants to change the sound. Therefore, he clicks on the first brown button below the soundboard so he can add new samples to the library of samples from the samples the Freesound API offers (screen 2). Back to the main page, he has to press for one or two seconds on the samples/ button if he wants to change the sound. Then he will have to click on the first button “Remplacer le son par un son dans la bibliothèque” to do that (screen 3). He will be able to change it now (screen 4).


19:49
70%
Soundboard
Trouver un son dans la librairie FreeSound
Enregistrer un sample avec son microphone


14:08
62%
Soundboard - Trouver un s...
Car

Car Wipers Interior

AJOUTER CE SAMPLE À LA LISTE
JOUER LE SAMPLE

CarEngine.wav

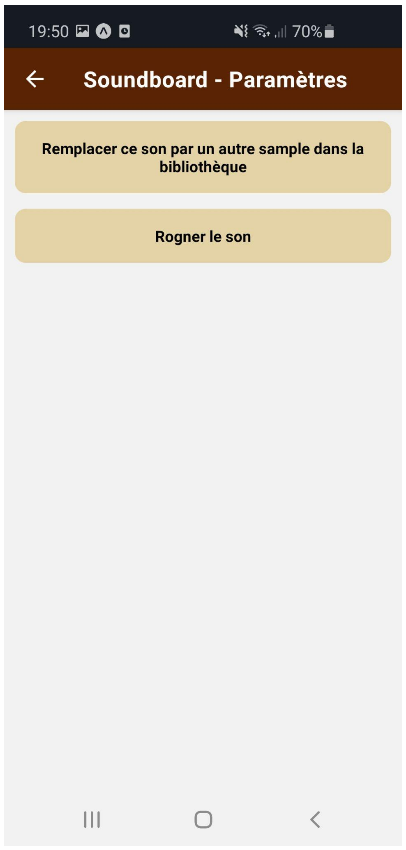
AJOUTER CE SAMPLE À LA LISTE
JOUER LE SAMPLE

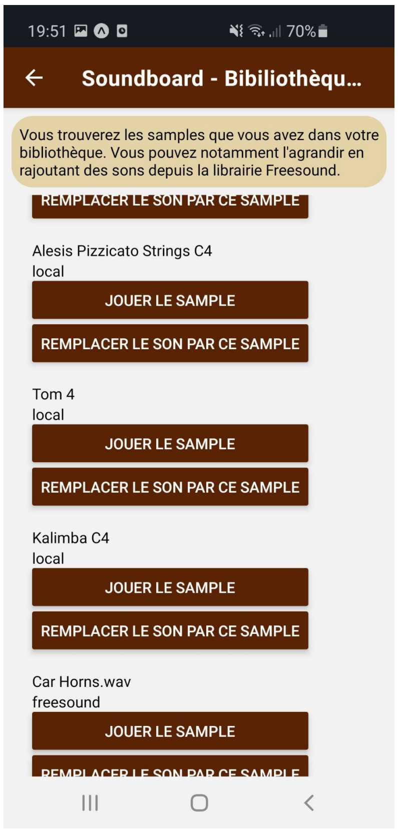
Car Starting and Accelerating

AJOUTER CE SAMPLE À LA LISTE
JOUER LE SAMPLE

RFX_Car Engine, Acelerator and Switch Off.wav

AJOUTER CE SAMPLE À LA LISTE
JOUER LE SAMPLE




19:51
70%
Soundboard - Bibiliothèqu...

Vous trouverez les samples que vous avez dans votre bibliothèque. Vous pouvez notamment l'agrandir en rajoutant des sons depuis la librairie Freesound.

REMPLACER LE SON PAR CE SAMPLE

Alesis Pizzicato Strings C4
local

JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

Tom 4
local

JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

Kalimba C4
local

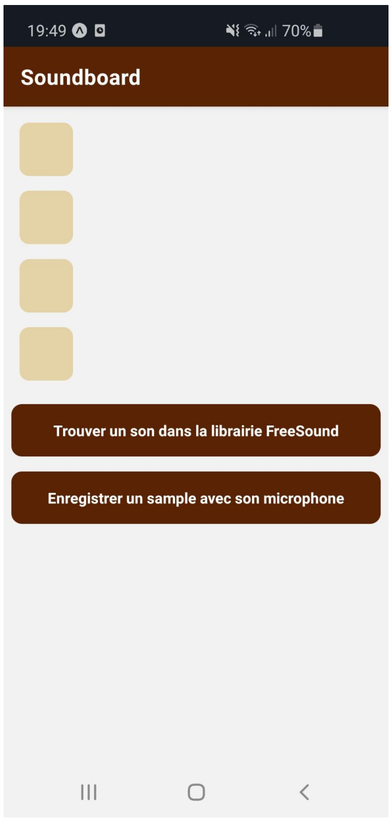
JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

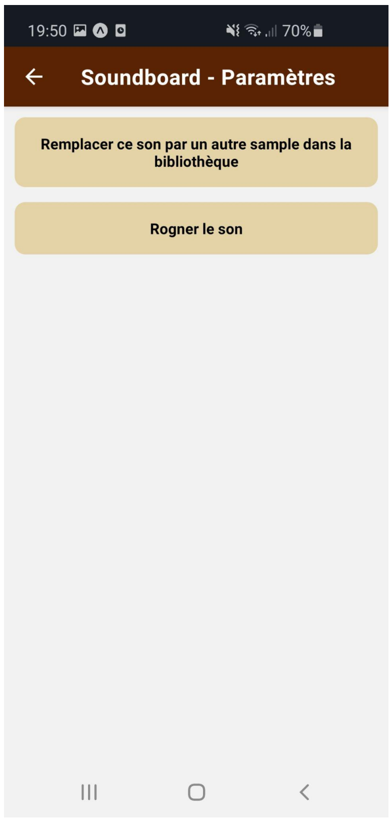
Car Horns.wav
freesound

JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

IV / Current state of the project

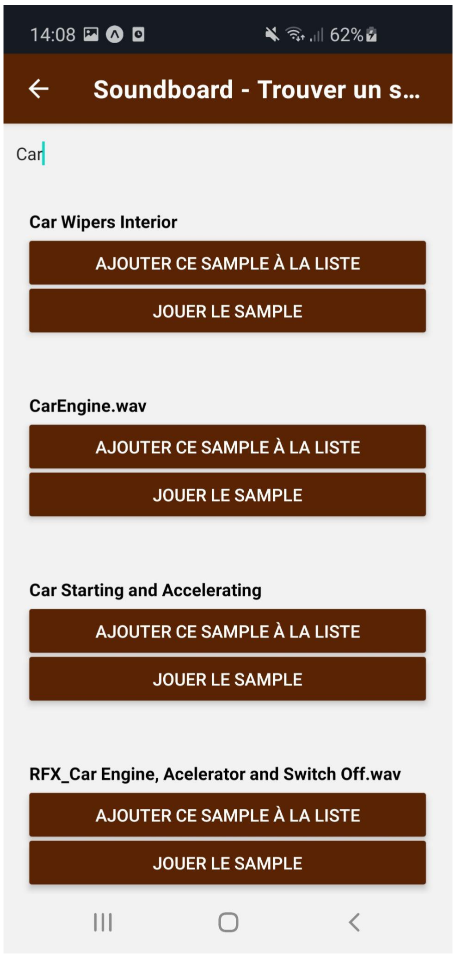
- Views & Redux store


19:49
70%
Soundboard
Trouver un son dans la librairie FreeSound
Enregistrer un sample avec son microphone



During the time that was given to me, I could code some features. The first thing I worked on was the view of the soundboard and the implementation of the redux store.

For instance, when we click on the buttons, we can play the sample. To fetch the samples, I only had to use my Redux store.
When you press continuously on the button, you will be able to see a menu that allows the user to change the sound or find a new one.



Car Wipers Interior

AJOUTER CE SAMPLE À LA LISTE

JOUER LE SAMPLE

CarEngine.wav

AJOUTER CE SAMPLE À LA LISTE

JOUER LE SAMPLE

Car Starting and Accelerating

AJOUTER CE SAMPLE À LA LISTE

JOUER LE SAMPLE

RFX_Car Engine, Acelerator and Switch Off.wav

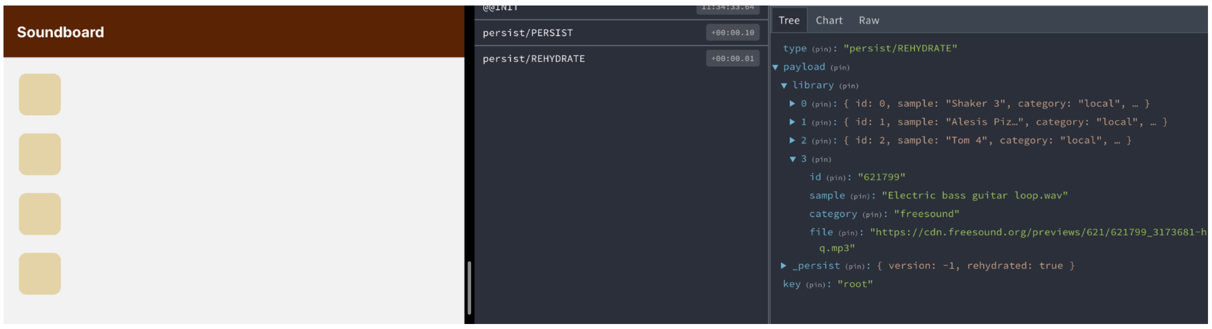
AJOUTER CE SAMPLE À LA LISTE

JOUER LE SAMPLE

This is the screen for the Freesound samples. When the user enters a keyword in the search bar, I make a request to the API (link to the website of the API). Then I display the results. I managed to play the sample so the user can listen to it before adding it to its library.

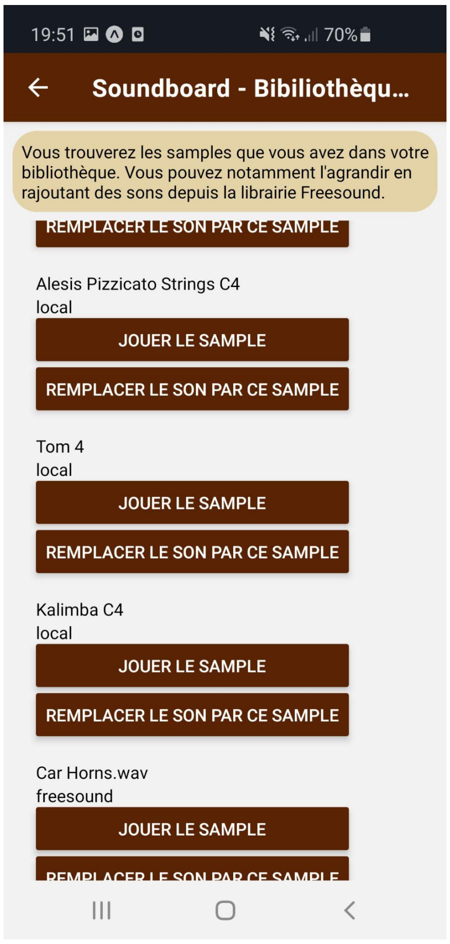
I also could add the sample to the redux store when the user clicks on the button to do that. To do so, I called a function that I created in the library slice.

Here is a little screenshot to see my redux state object once I added a new sample to the library slice.



```
@@INIT                     11:34:33.64
persist/PERSIST            +00:00.10
persist/REHYDRATE          +00:00.01
```

```
type (pin): "persist/REHYDRATE"
▼ payload (pin)
  ▼ library (pin)
    ▶ 0 (pin): { id: 0, sample: "Shaker 3", category: "local", … }
    ▶ 1 (pin): { id: 1, sample: "Alesis Piz…", category: "local", … }
    ▶ 2 (pin): { id: 2, sample: "Tom 4", category: "local", … }
    ▼ 3 (pin)
        id (pin): "621799"
        sample (pin): "Electric bass guitar loop.wav"
        category (pin): "freesound"
        file (pin): "https://cdn.freesound.org/previews/621/621799_3173681-h
          q.mp3"
  ▶ _persist (pin): { version: -1, rehydrated: true }
key (pin): "root"
```


19:51 70%

Soundboard - Bibiliothèqu…

Vous trouverez les samples que vous avez dans votre bibliothèque. Vous pouvez notamment l'agrandir en rajoutant des sons depuis la librairie Freesound.

REMPLACER LE SON PAR CE SAMPLE

Alesis Pizzicato Strings C4
local
JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

Tom 4
local
JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

Kalimba C4
local
JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

Car Horns.wav
freesound
JOUER LE SAMPLE
REMPLACER LE SON PAR CE SAMPLE

I could also display the samples of the user in the library and this is the view I created for it.

I managed to replace the sample of one button on the soundboard: after testing it both on my phone and laptop, I realized it only worked on my laptop.
On both devices, I can replace the sample in the redux store however it only plays the new sample on my laptop.
It would be great if you can test the application on your computer and phone if you have the time to check if it works on your laptop also at least.

- Persisted storage

I also could implement Redux Persist to my project: thanks to this library, I can save the data I want to keep even after I reload the page or relaunch the app.

Therefore, if the user adds a new sample in his library hence in the redux state object, he will be able to find it once he comes back to the application.