Soundplanes

Architecture and Integration of Software Systems

Software Engineering degree

Course year 2019/20

Georgeo Laurentiu Bogdan (geobog@alum.us.es)
Guillermo Diz Gil (guidizgil@alum.us.es)
Carmen María Muñoz Pérez (carmunper1@alum.us.es)
Francisco Rodríguez Pérez (frarodper4@alum.us.es)

Form teacher: Javier Troya
Group number: 1

Application link: https://soundplanes.appengine.com/
Link of the project on Github, projETSII or similar: https://github.com/GuilleX7/Soundplanes

VERSION HISTORY

Date	Version	Details	Participants
14/03/2014	1.0	- Includes introduction, prototypes of the user interfaces and UML diagrams of components and deployment	George Laurentiu Bogdan Guillermo Diz Gil Carmen Mª Muñoz Pérez Francisco Rodríguez Pérez
03/05/2014	2.0	- Added class and sequence diagrams, code updated, preliminary version of the application.	George Laurentiu Bogdan Guillermo Diz Gil Carmen Mª Muñoz Pérez Francisco Rodríguez Pérez
24/05/2020	3.0	- Added API, unit tests and integration test along their documentation. Final version of the application deployed.	George Laurentiu Bogdan Guillermo Diz Gil Carmen Mª Muñoz Pérez Francisco Rodríguez Pérez

Index

1 Introduction

Today, people are increasingly interested in learning more about other cultures. Thanks to technologies that are continuously growing, it is becoming easier to establish this communication. On the other hand, music is a key element in our society and unites people from all over the world. Even so, with the exception of very international artists, we are not aware of the musical culture of countries other than our own. Since music is such an important cultural element in each country and in people, we intend to carry out a project that consists of offering an innovative and educational service in the form of an international online radio.

Thus, Soundplanes wants to offer an interactive online radio service with the possibility of chatting with people from anywhere in the world. It will give its users the opportunity to listen to the most listened songs from each country, as well as to offer their own playlist. In this way, users will have the opportunity to learn new songs, as well as meet many people from around the world.

1.1 Integrated applications

We will be using the following applications:

- Youtube: Youtube web player will be our music player. We will also use its REST API for fetching some metadata.
- Spotify: we will be using Spotify as a music search site. The user will be able to search for any existing song in Spotify and play it.
- Genius: will be used to fetch song's lyrics.
- Leaflet: will be used for displaying interactive maps in the web browser. Used in combination with *OpenStreetMap*
- Facebook: we will allow the user to log in with his Facebook social account and post what music are they listening to at any time.
- Google Geocoding: will allow us to convert addresses into geographic coordinates, so we can position the player in the map in case we were unable to geolocate him/her.
- Instant IRC Chat: will allow us to create different chat channels, one for each airport, so users can chat among themselves.

Application name	URL API documentation
Youtube	https://developers.google.com/youtube/iframe_api_reference?hl=es https://developers.google.com/youtube/v3/docs
Spotify	https://developer.spotify.com/documentation/web-api/
Genius	https://docs.genius.com/
Leaflet	https://leafletjs.com/reference-1.6.0.html
Facebook	https://developers.facebook.com/docs/javascript
Google Geocoding	https://developers.google.com/maps/documentation/geocoding/intro?hl=es-419
Instant IRC Chat	https://documenter.getpostman.com/view/6300420/SzmZdgNU?version=latest

1.2 Project development

We are thinking about all the different applications that we could integrate in our final application. Some of these may not work as expected and be removed, or we could change our visions about what do we want our application to be.

As of April 27, we know all the APIs we are using in the project, and we have obtained all the corresponding API keys in order to consume their services. We plan to keep the application simple but powerful and elegant. We are putting a lot of emphasis in the user experience.

As of May 3, we are thinking about the possibilities of the Facebook API. We already have included it for signing in, but we want to explore the Facebook universe in a deeper way.

As of May 24, we have already built the entire application. We have kept Facebook as a social login service. The initial idea of a synchronised track in every airport was discarded. Plane movement has been real-time

synchronised, instead. A lot of effort has been put on user interface, which has improved a lot, as it is responsive and well displayed on almost every device no matter screen size.

2 User Interface Prototypes

Our application will work as a single-page application. This means we will display all our views dynamically, without the loading of additional pages. We expect this to improve the user experience.

2.1 Landing page view

A first view of the page, where the project is described to the user.

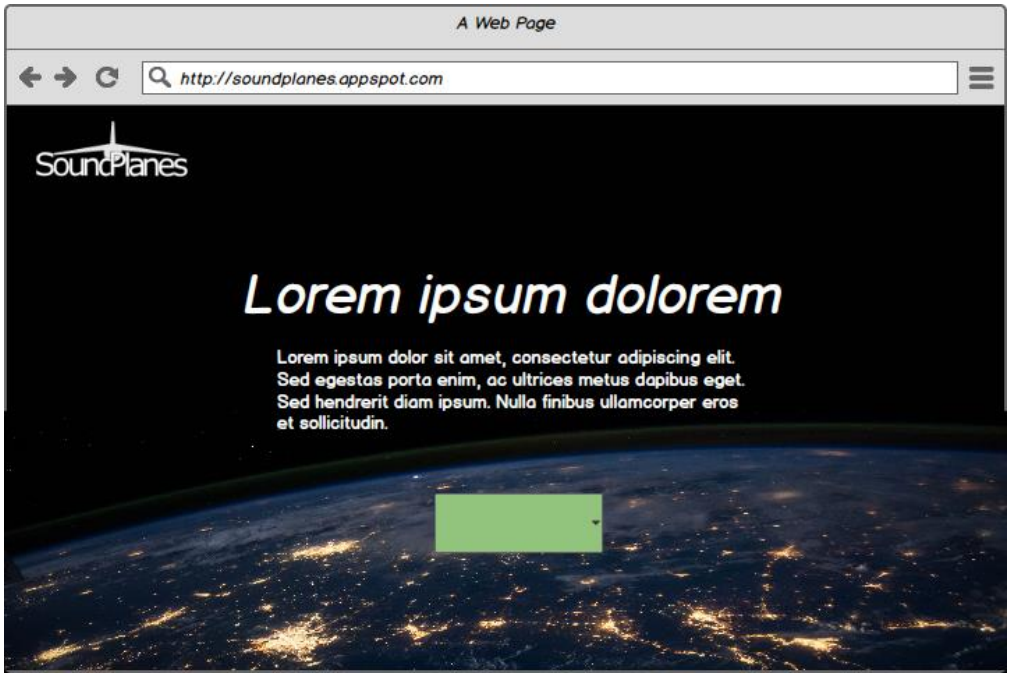


FIGURE 1. LANDING PAGE VIEW'S USER INTERFACE PROTOTYPE

2.2 User register view

User is requested his nickname to be displayed in-game, after that it has two options, to allow the use of user's location in order to improve his experience or to choose the location manually, this is where its base will be located.

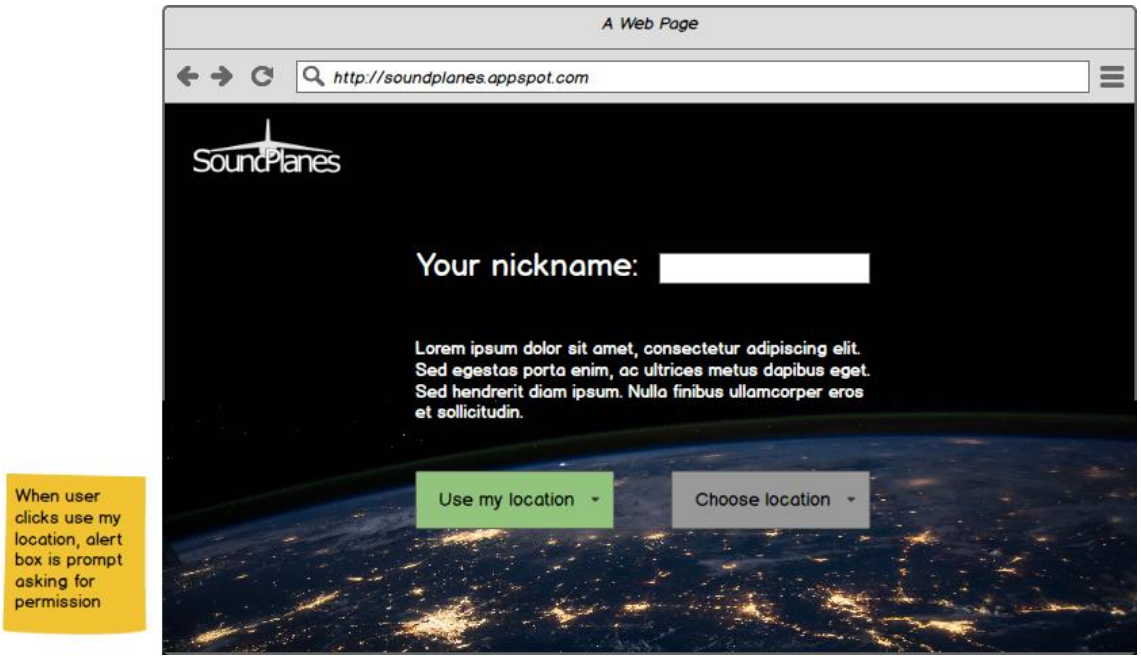


FIGURE 2. SETUP VIEW'S USER INTERFACE PROTOTYPE

2.3 Location view

If user want to choose location manually, it'll show the country from where he is visiting our site, and he will pick the region of the country. If he wants to use its location automatically, he can go back.

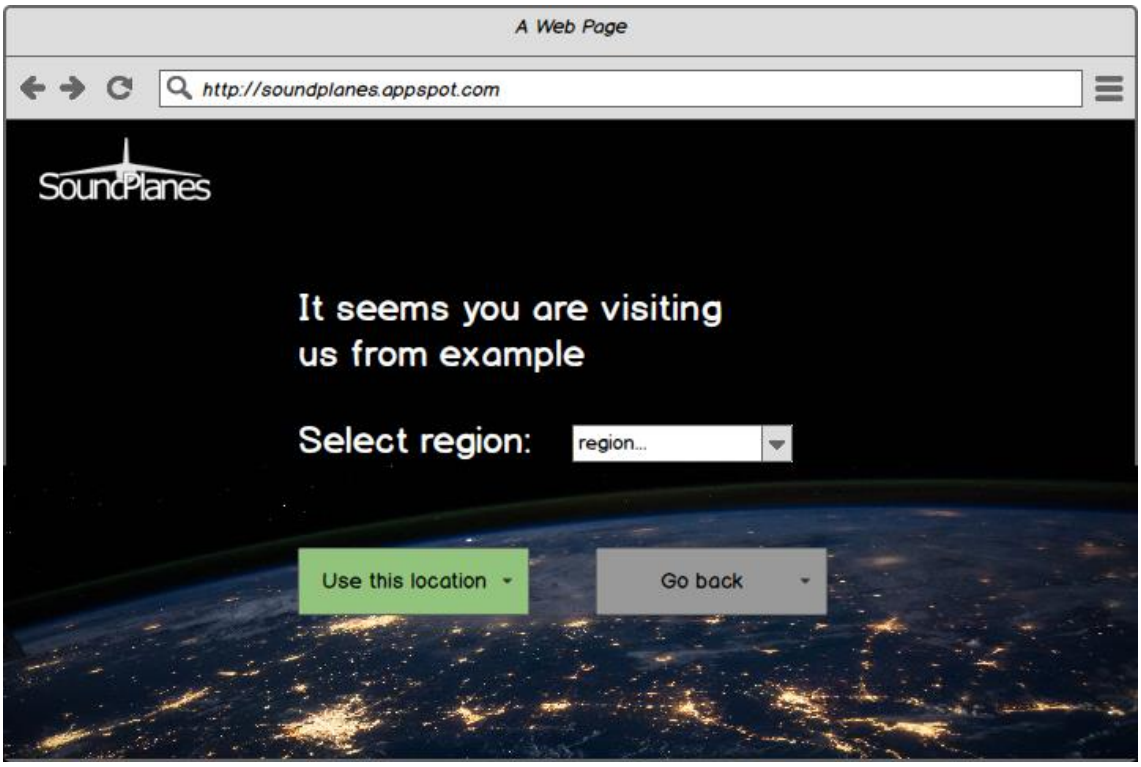


FIGURE 3. LOCATION VIEW'S USER PROTOTYPE INTERFACE

2.4 Main view

This will be the main view of the player, the region where the player is will be displayed, the plane is user's position and he can see nearby airports.

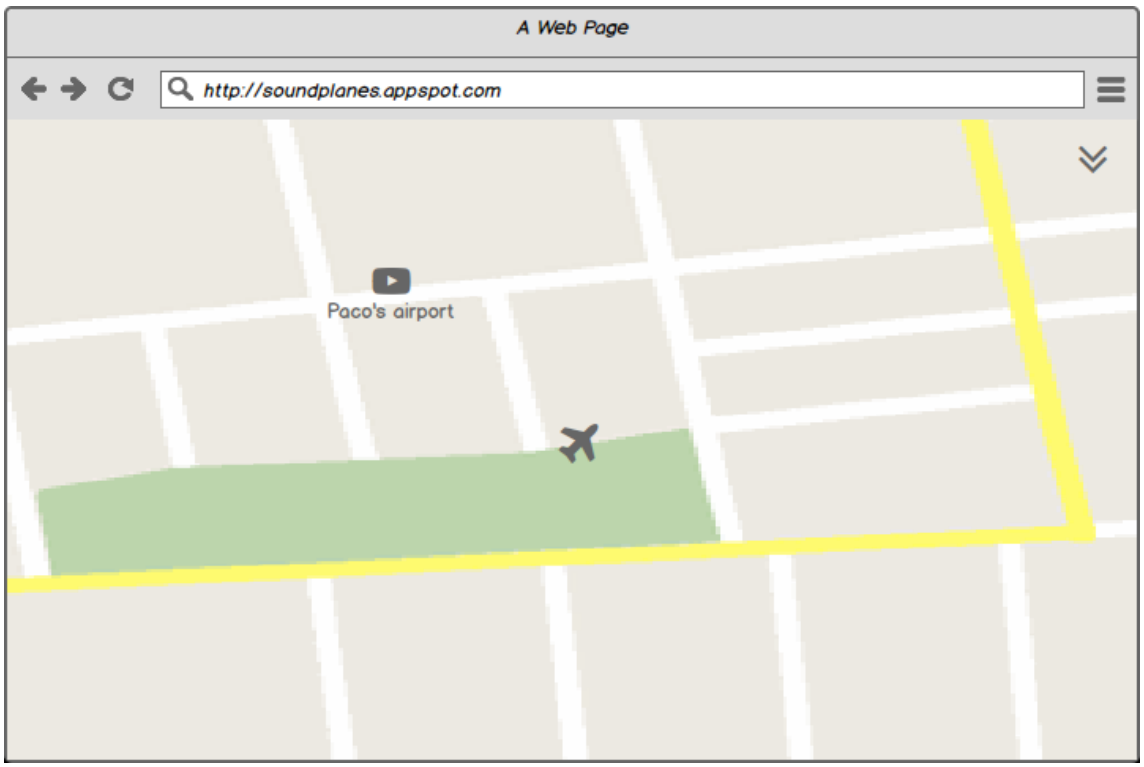


FIGURE 4. MAIN VIEW'S USER PROTOTYPE INTERFACE

2.5 Profile menu view

In this tab it is displayed user information, where he can change his nickname or link his account with Spotify or Facebook. You can also enter your airport's menu from here.

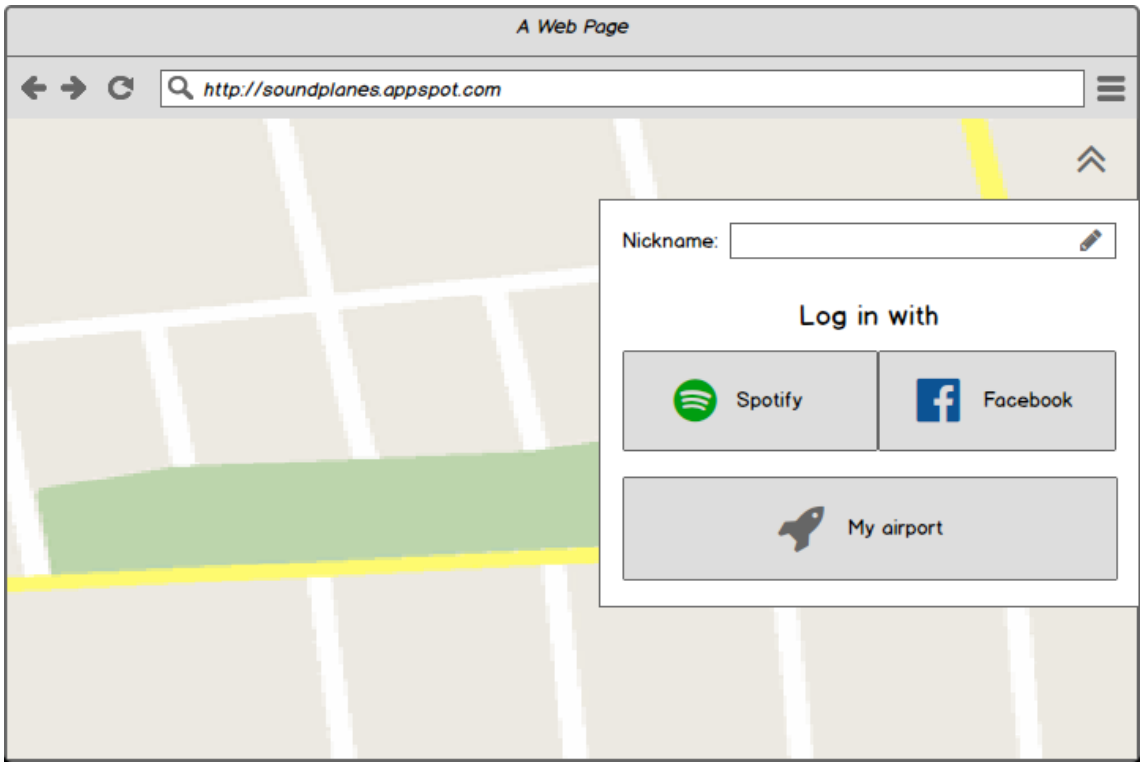


FIGURE 5. PROFILE MENU VIEW'S USER INTERFACE PROTOTYPE

2.6 Airport menu view

Here is where it's displayed user's airport information. User can change airport's name, see the number of visitors that are currently in the airport, and the total number of visitors overall. User can also change the actual playlist of the base or delete the airport.

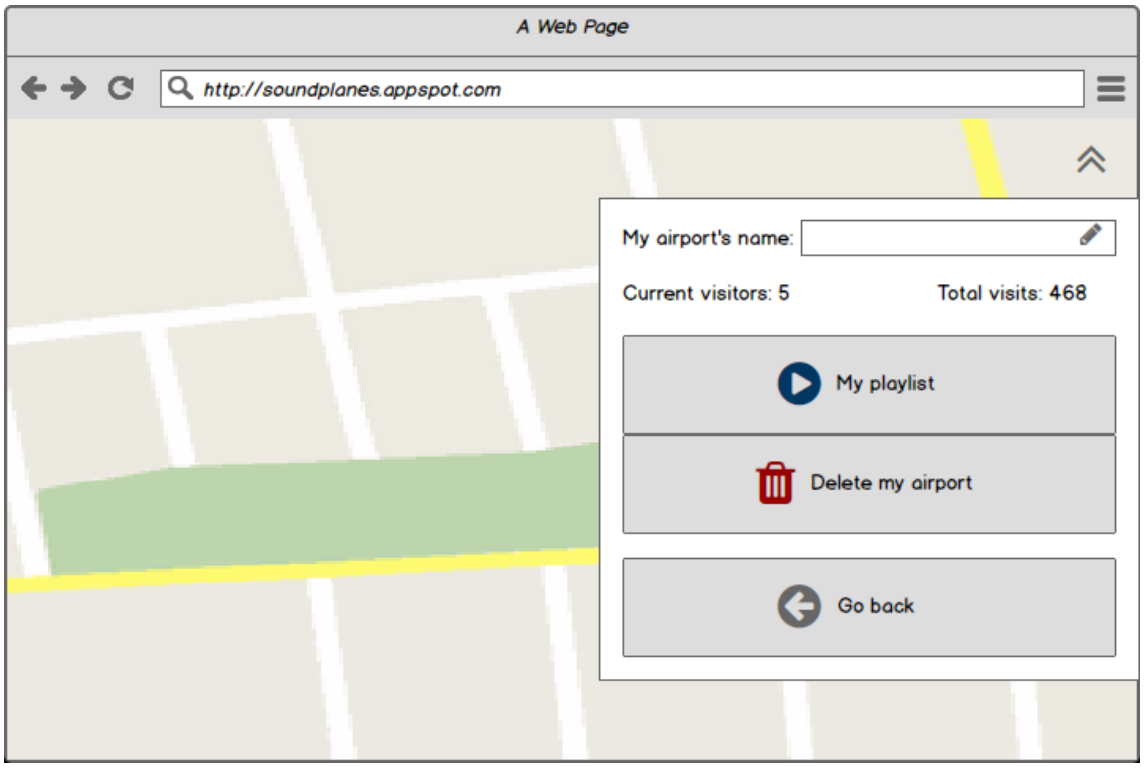


FIGURE 6. BASE MENU VIEW'S USER INTERFACE PROTOTYPE

2.7 Airport chat view

Here is the chat of the airport where users can chat with each other.

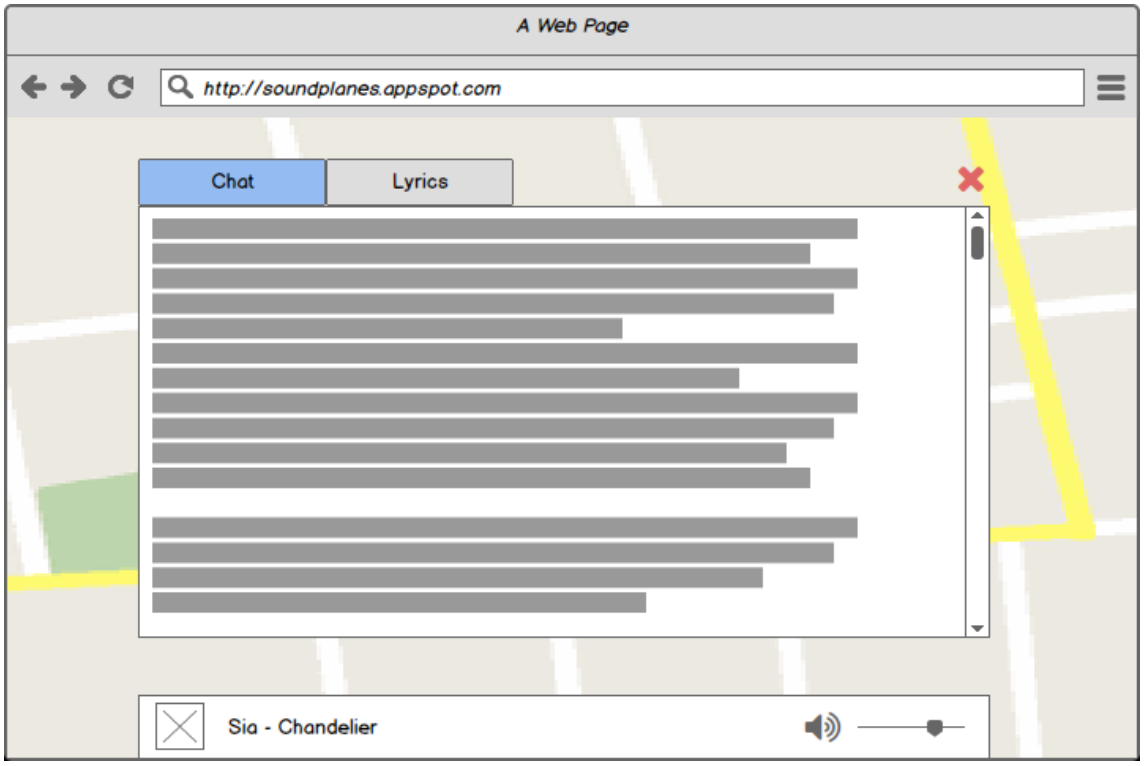


FIGURE 7. AIRPORT CHAT VIEW'S USER INTERFACE PROTOTYPE

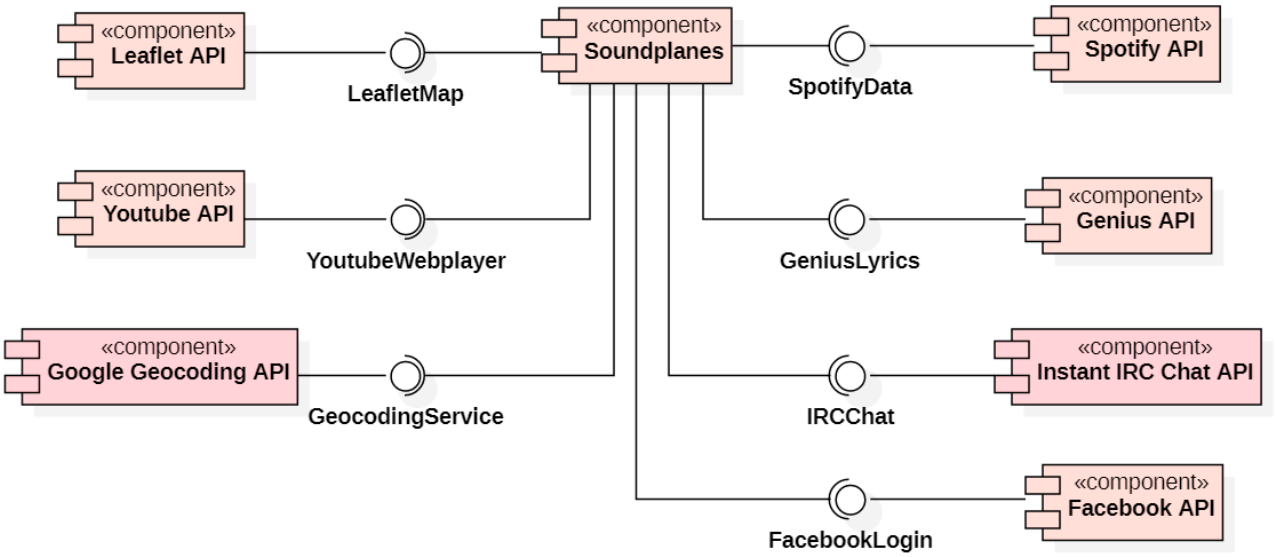
2.8 Airport lyrics view

Lyrics of the song that is currently playing in the airport.

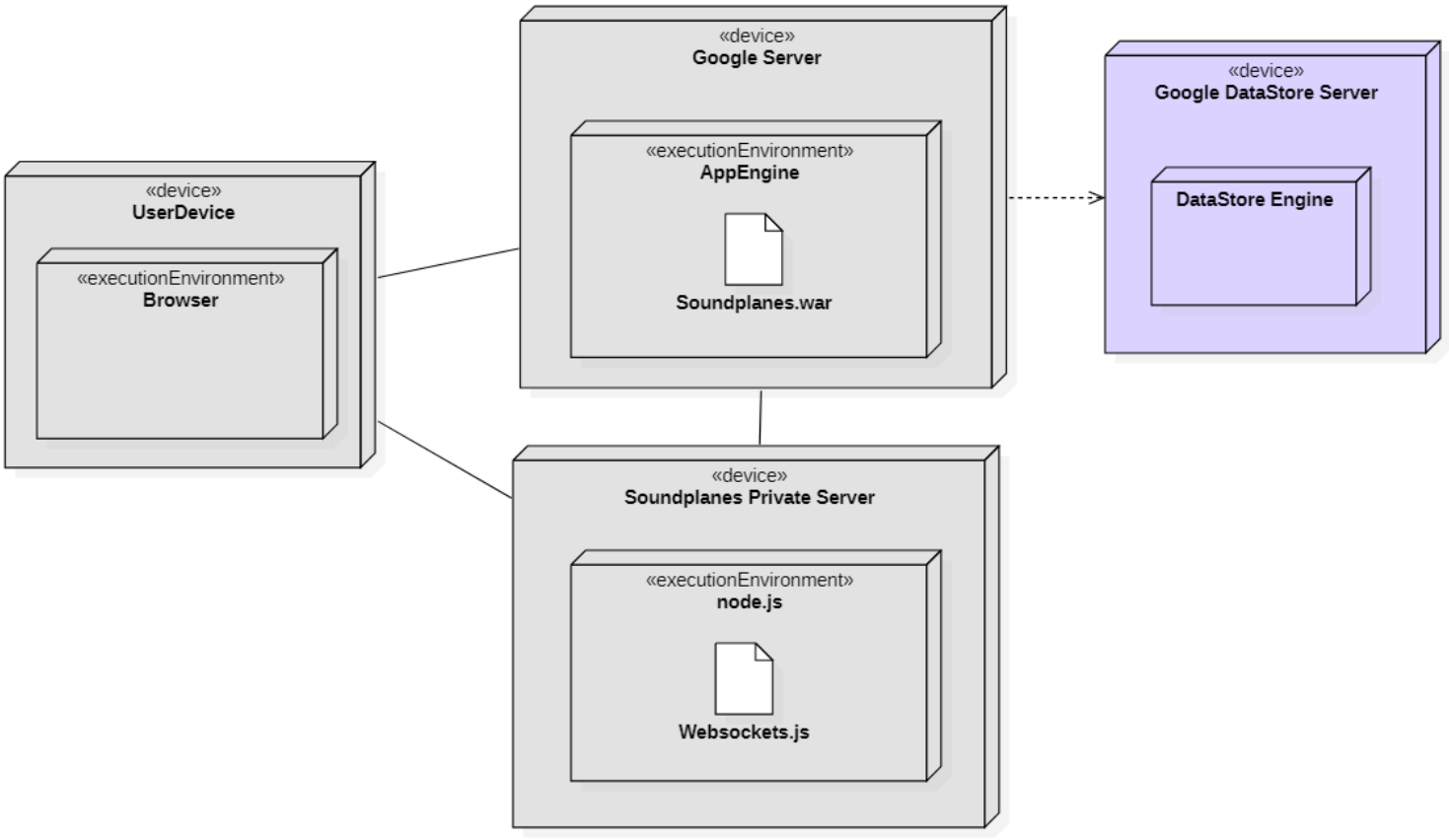
FIGURE 8. AIRPORT LYRICS VIEW'S USER INTERFACE PROTOTYPE

3 Architecture

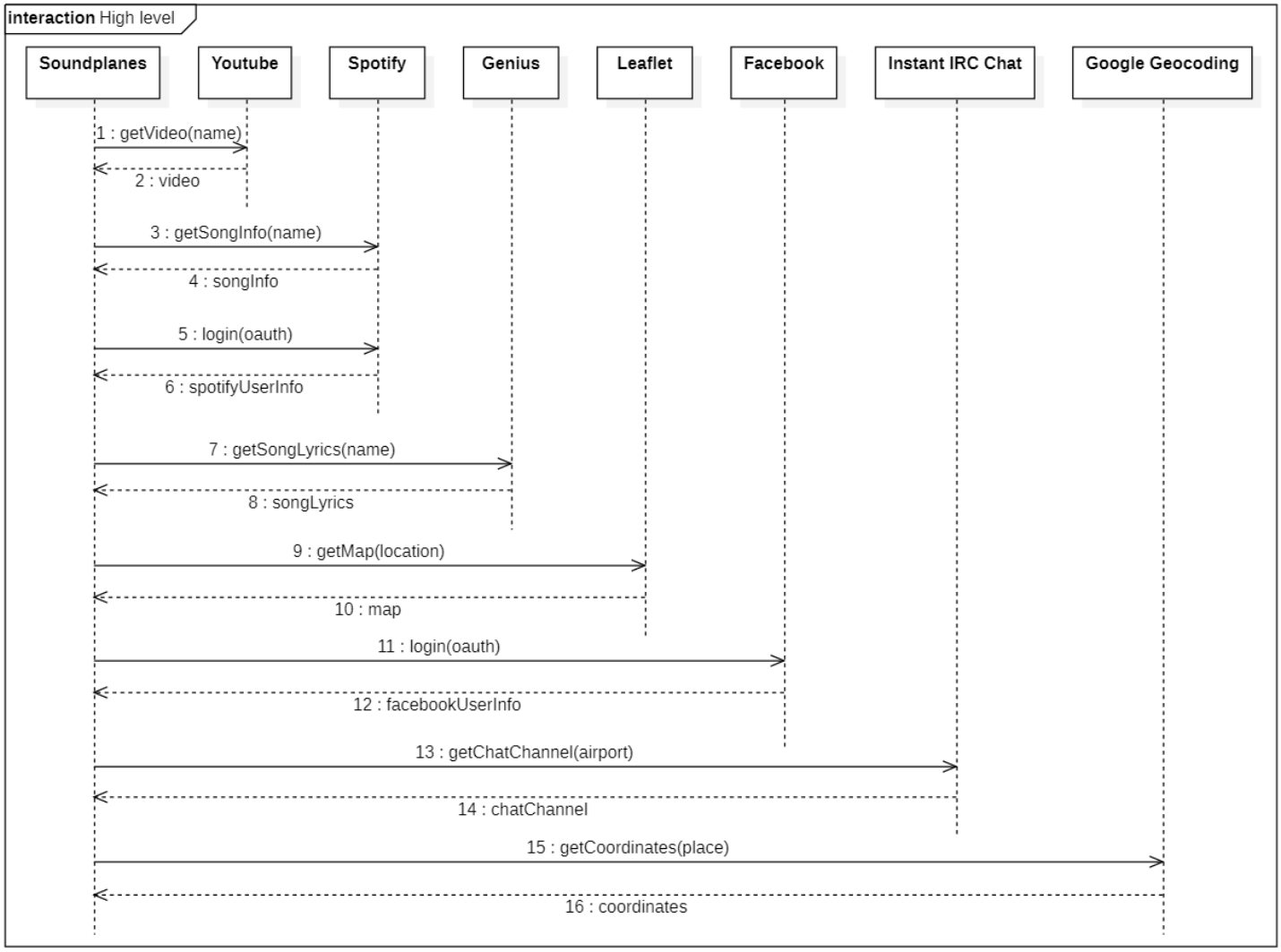
3.1 Component diagram



3.2 Deployment diagram

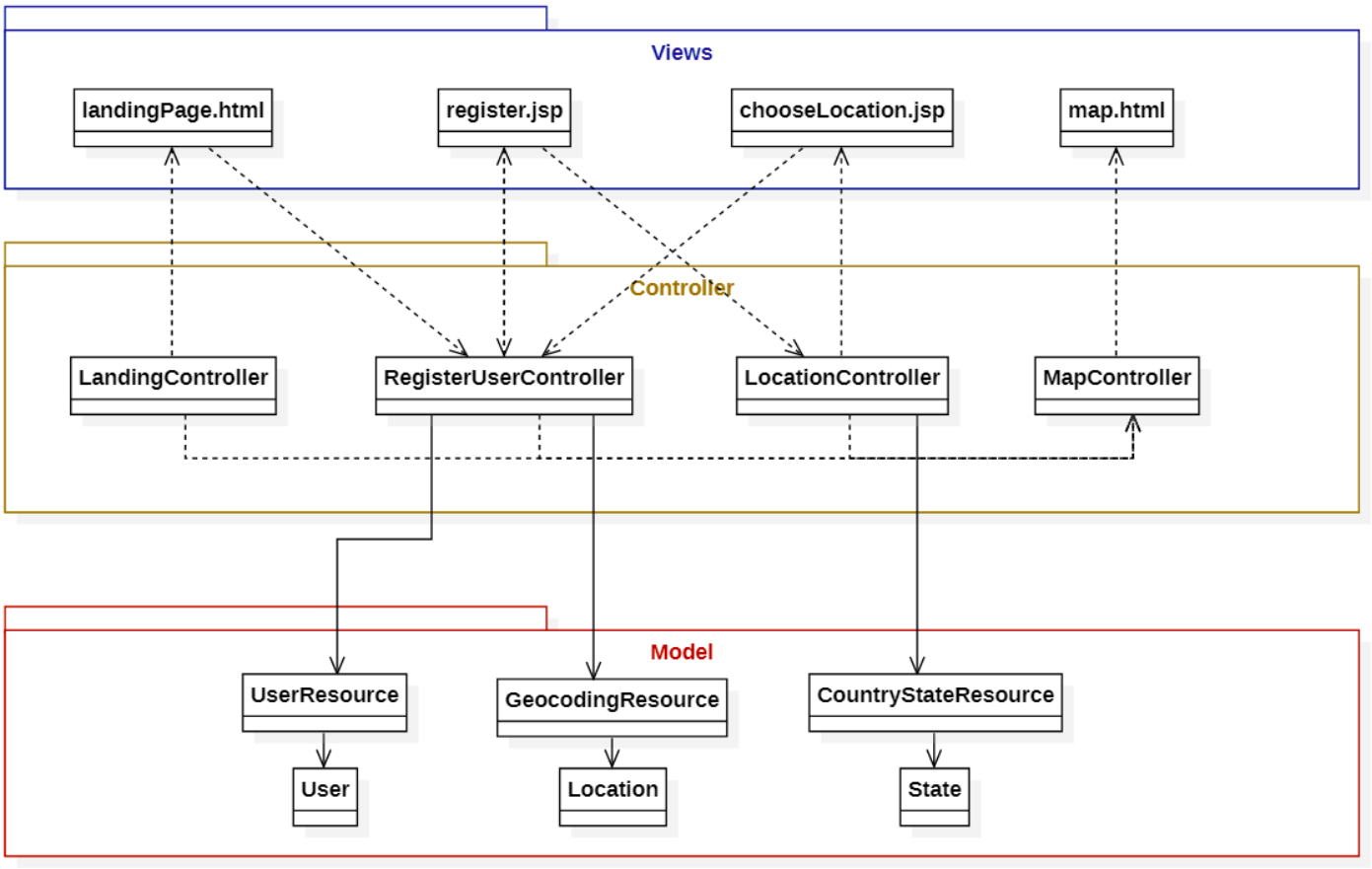


3.3 High-level sequence diagram

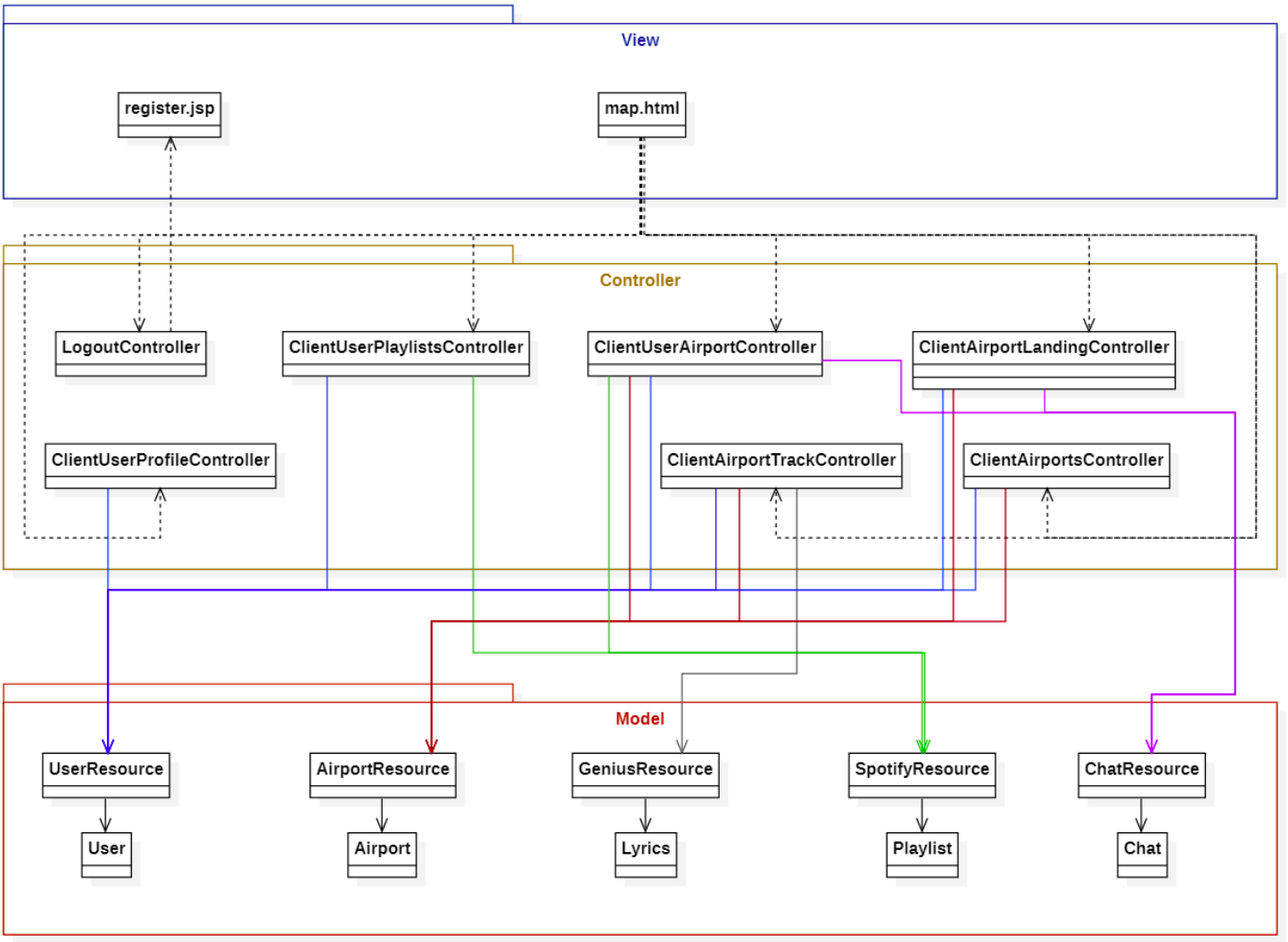


3.4 Class diagram

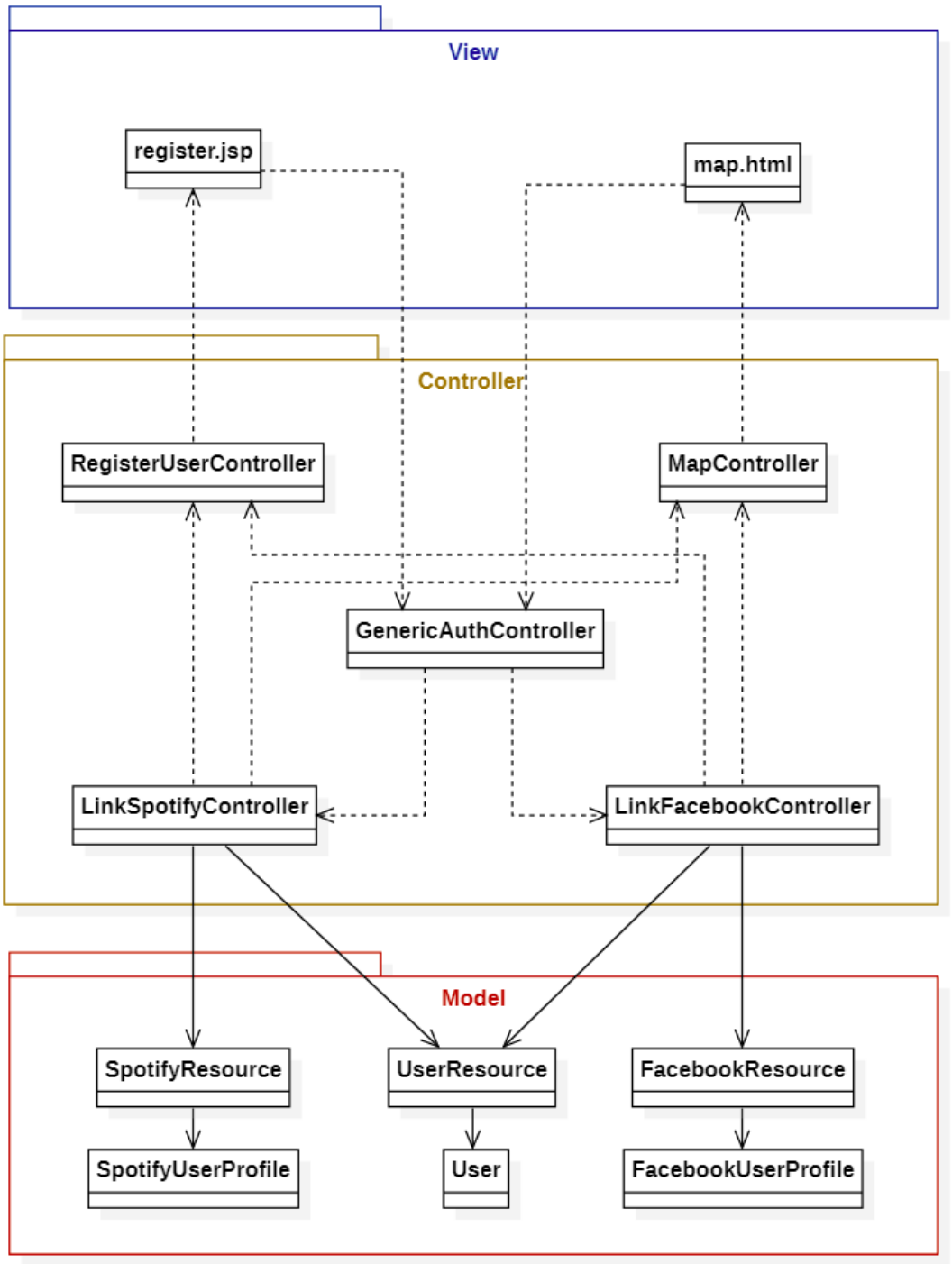
3.4.1 Landing/sign in class diagram



3.4.2 Map class diagram



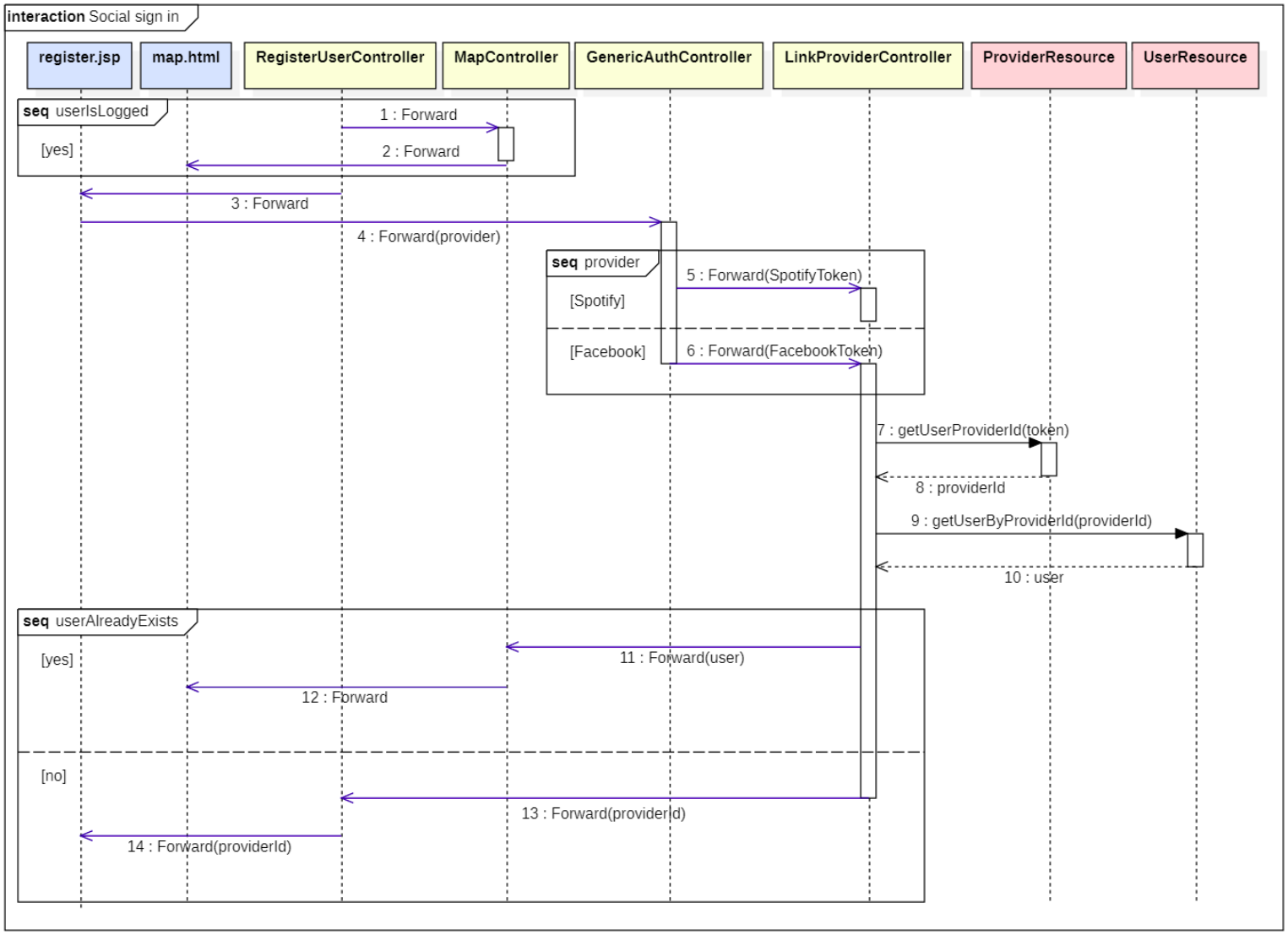
3.4.3 Social sign in/linking class diagram



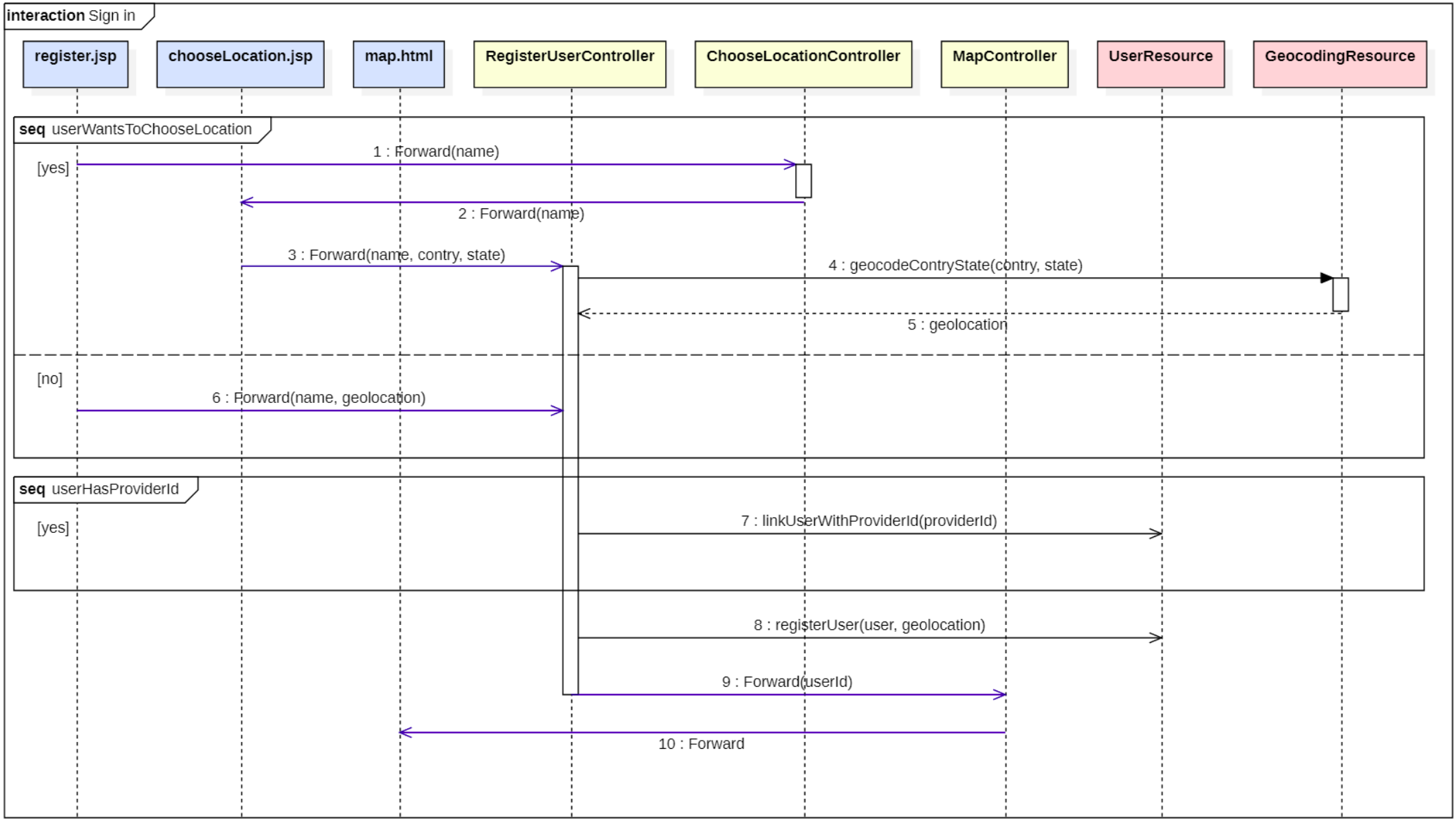
3.5 Sequence diagram

3.5.1 Social sign in sequence diagram

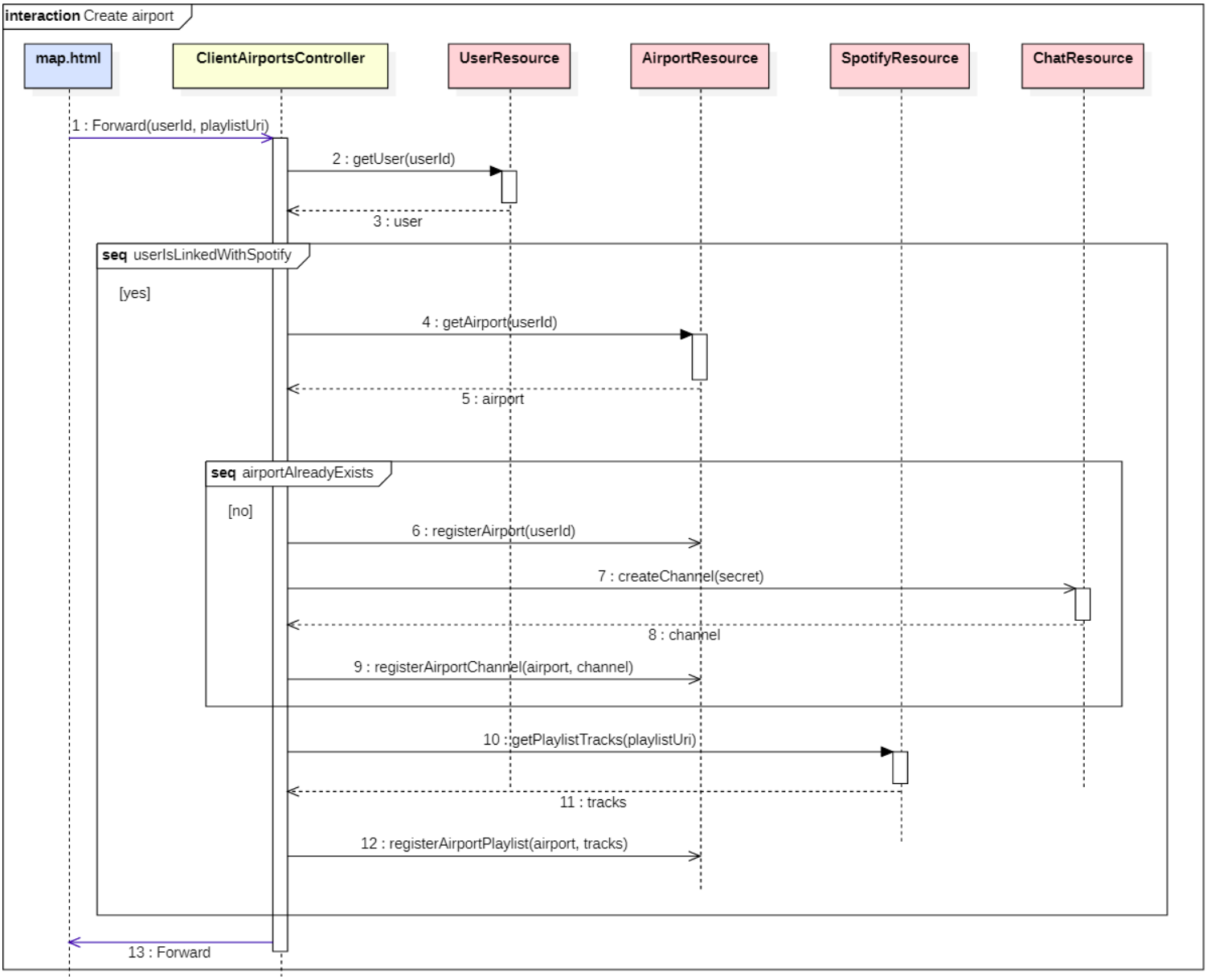
Some generalizations have been made in this diagram. “LinkProvider” and “ProviderResource” refers to both tuples “LinkSpotify”, “SpotifyResource” and “LinkFacebook”, “FacebookResource”. Their behaviour is the same, this is just a short name for those classes.



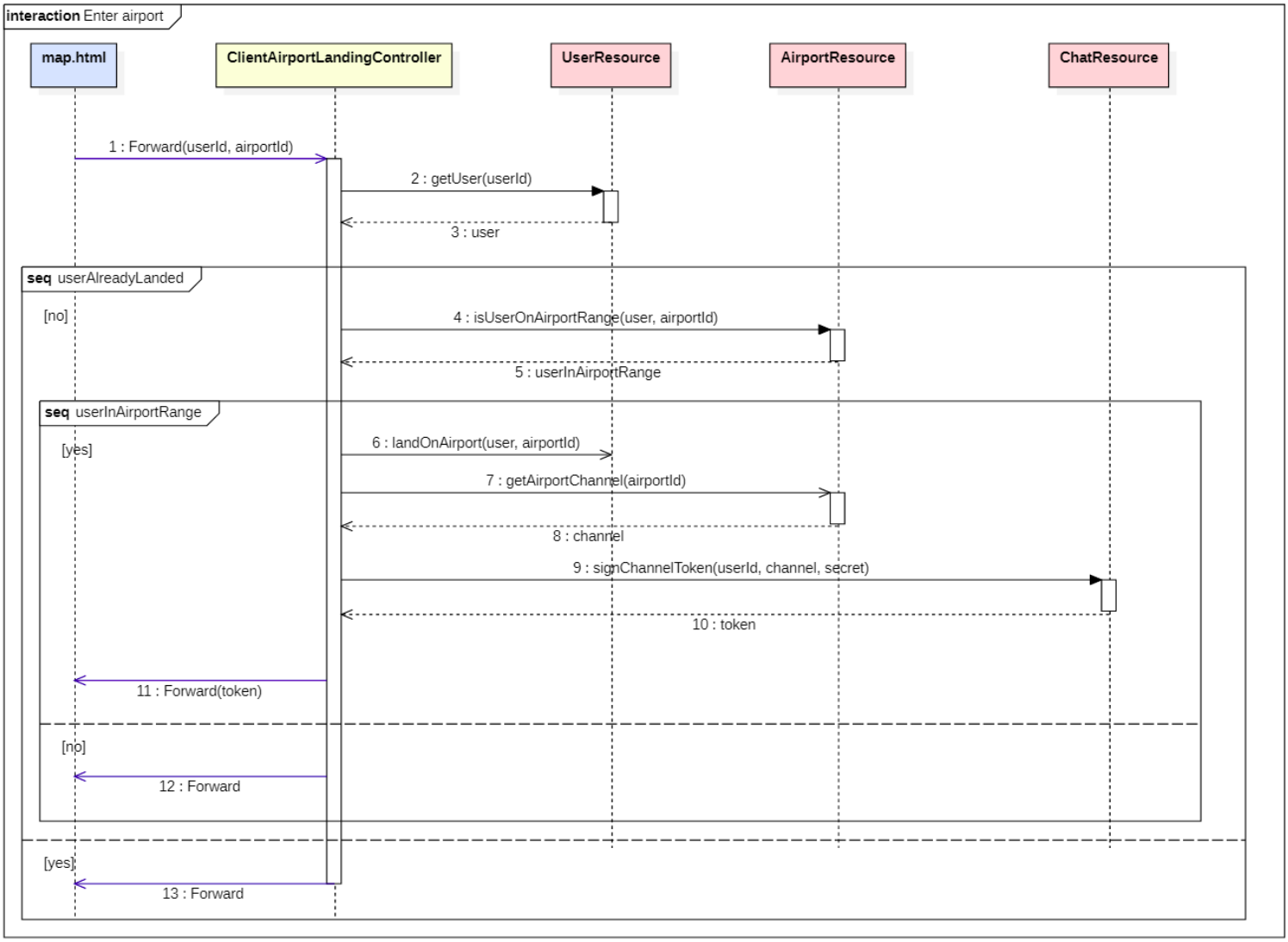
3.5.2 Sign in sequence diagram



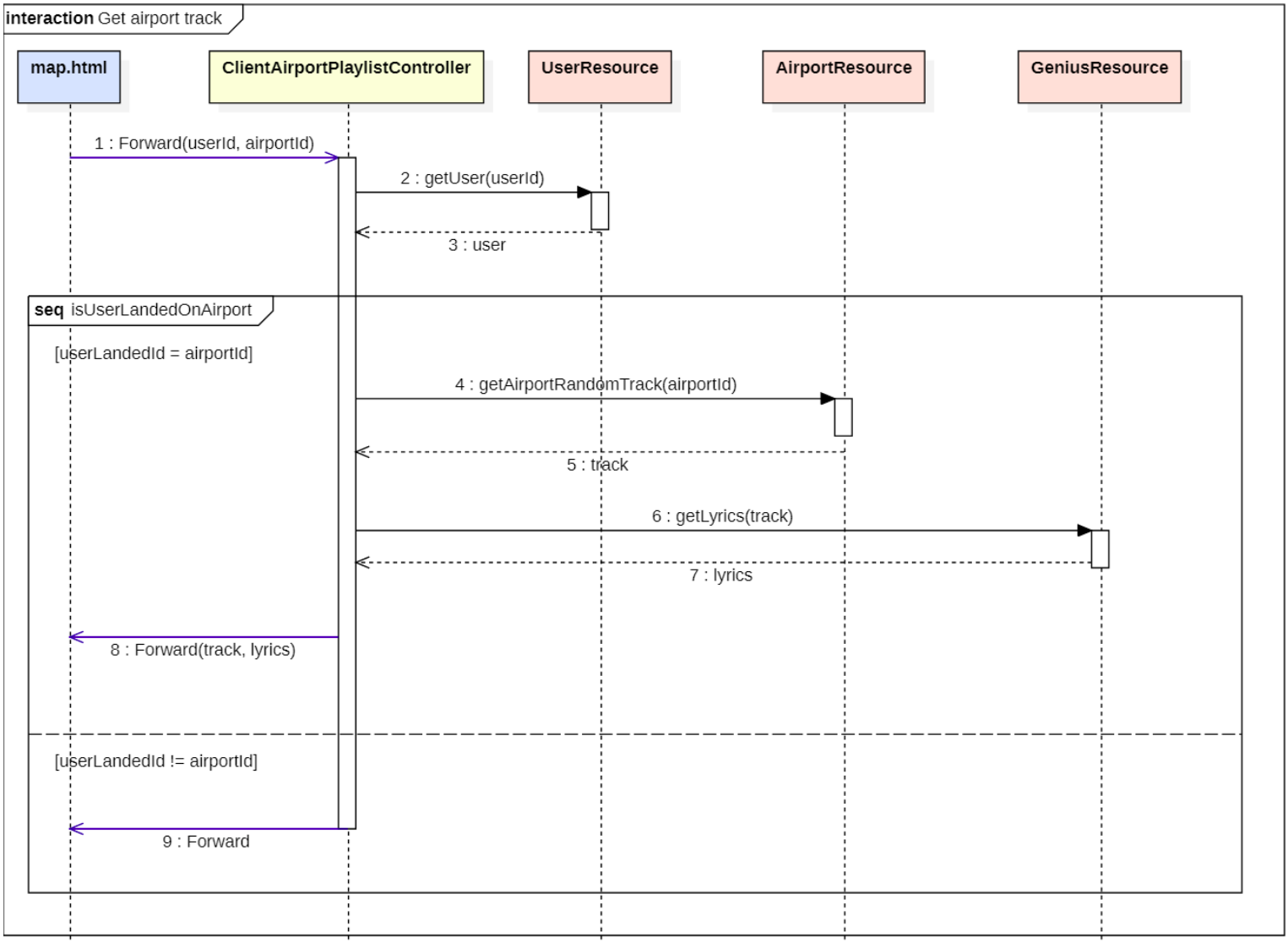
3.5.3 Create airport sequence diagram



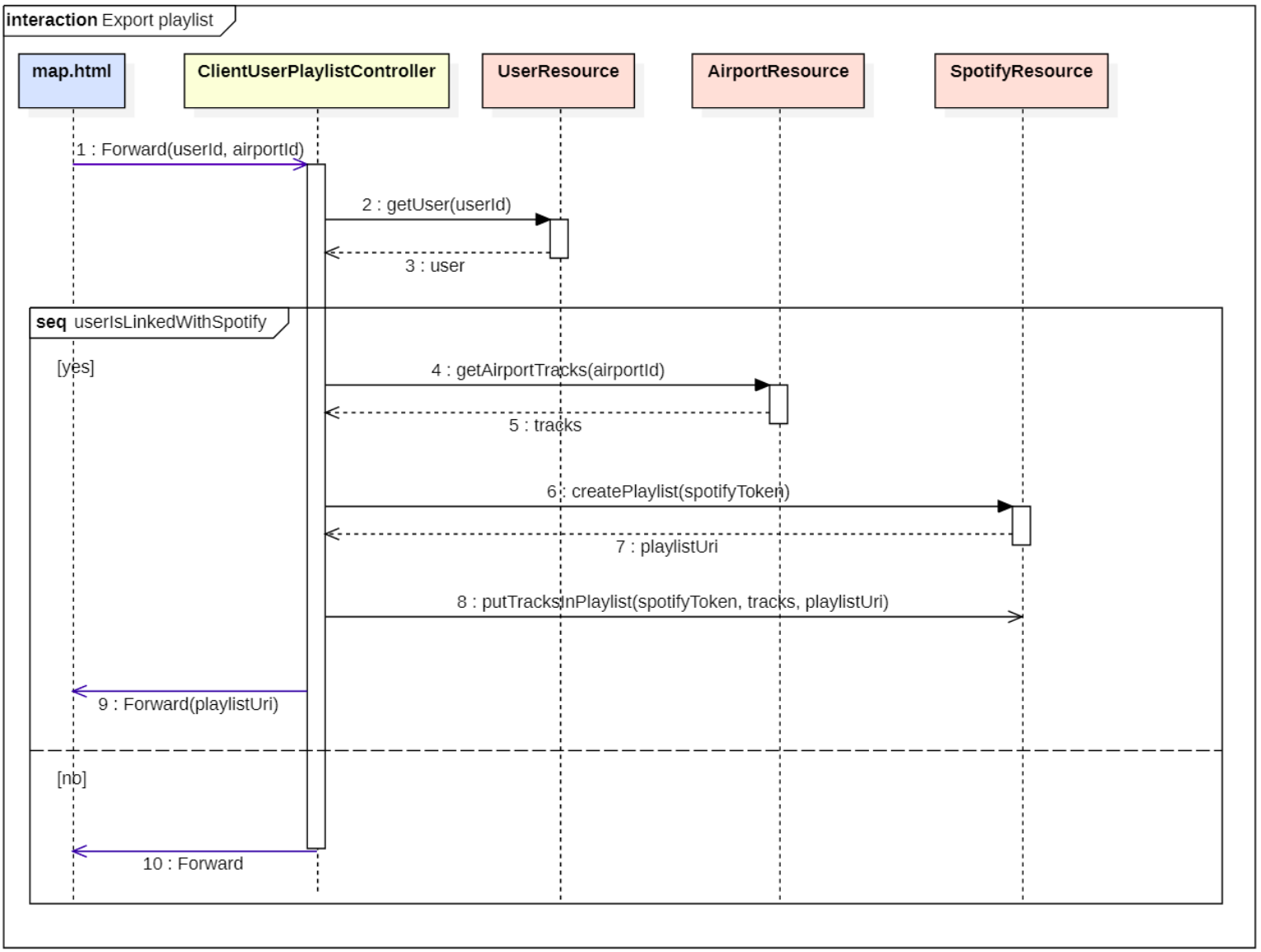
3.5.4 Enter airport sequence diagram



3.5.5 Get airport track sequence diagram



3.5.6 Export playlist sequence diagram



4 Implementation

Soundplanes aims to be an innovative, modern and simple way to gather people around the world. Its because of this that we needed to take our application to the next level.

The core of Soundplanes is made entirely with Java. It runs on Google AppEngine servers. However, instead of being a generic, *non-stated* mashup, we have decided to use Google DataStore API, in order to save registered users and their data. Objectify makes the hard work for us: local singleton repositories have been replaced with static objectify calls that communicates with the low-level DataStore API in order to save and retrieve data in a transparent way for our core.

We have made use of Bootstrap and JQuery for the client side. This makes our web design fully responsive. Our client will try their best to offer the best user experience using the available browser features, no matter the device.

As the nature of our data is dynamic (moving around a map, chatting…), we used websockets in the client side. These websockets opens a connection with the API servers.

5 Tests

We have followed a bottom-up strategy because it was more convenient for us, as we started coding the different API integrations. We used drivers at early stage, so different components could be tested independently with unit tests. Finally, we tested the complete flow with integration tests.

5.1 Unit tests

Summary	
Total number of tests performed	31
Number of automated tests	25 (80%)

ID	AirportResource_Test
Description	Check all CRUD (Create Read Update Delete) methods of the AirportResource are valid.
Input	A valid user uuid
Expected output	Airport and AirportPlaylist are created, read and deleted successfully.
Result	Success
Automated	Yes

ID	CountryStatesResource_Test
Description	Check if states are loaded successfully
Input	Country Spain and invalid country
Expected output	Return states of Spain and 0 states for invalid country
Result	Success
Automated	Yes

ID	GeniusResource_Test
Description	Retrieves the lyrics of a song
Input	A song name
Expected output	The corresponding song lyrics
Result	Success
Automated	Yes

ID	GeocodingResource_Test
Description	Returns the geolocation of an address
Input	An address
Expected output	A geolocation object corresponding to the address
Result	Success
Automated	Yes

ID	IrcChatResource_Test
Description	Creates a channel and retrieves valid tokens for an user
Input	A secret key and user information
Expected output	A channel and a valid token to connect to that channel
Result	Success
Automated	Yes

ID	SpotifyResource_Test
Description	Check all CRUD (Create Read Update Delete) methods of the SpotifyResource are valid.
Input	An user token
Expected output	Playlist of that user is successfully created, read, updated and deleted
Result	Success
Automated	No (requires updating user token)

ID	FacebookResource_Test
Description	Return the profile of a given user
Input	An user token
Expected output	The profile of the given user
Result	Success
Automated	No (requires updating user token)

ID	UserResource_Test
Description	Check all CRUD (Create Read Update Delete) methods of the UserResource are valid.
Input	An uuid for the user
Expected output	User is successfully created, read, updated and deleted
Result	Success
Automated	Yes

5.2 Integration tests

Summary	
Total number of tests performed	2
Number of automated tests	2 (100%)

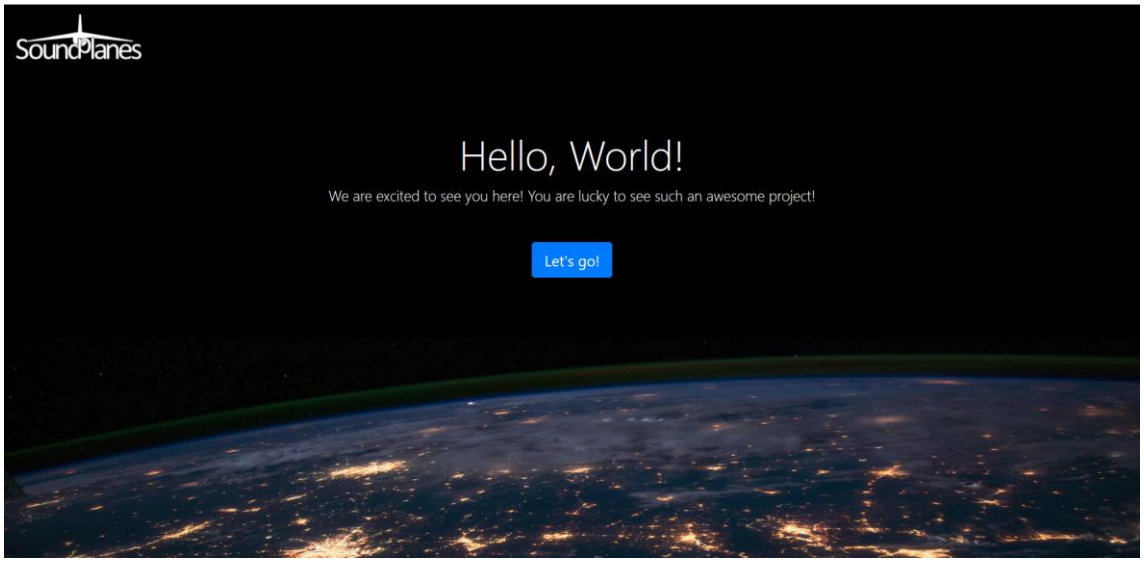
ID	UserFlow_IntegrationTest
Description	Check that user is registered successfully and is unable to create an airport until he/she logs in with Spotify
Input	User's name and location
Expected output	User is successfully registered and unable to create an airport
Result	Success
Automated	Yes

ID	ChatFlow_IntegrationTest
Description	Check the complete flow for a user that registers in the application and enters an airport, obtaining a chat token for the airport's channel successfully
Input	User's name and location
Expected output	User is successfully registered, able to create and to land on an airport and able to get a valid chat token
Result	Success
Automated	Yes

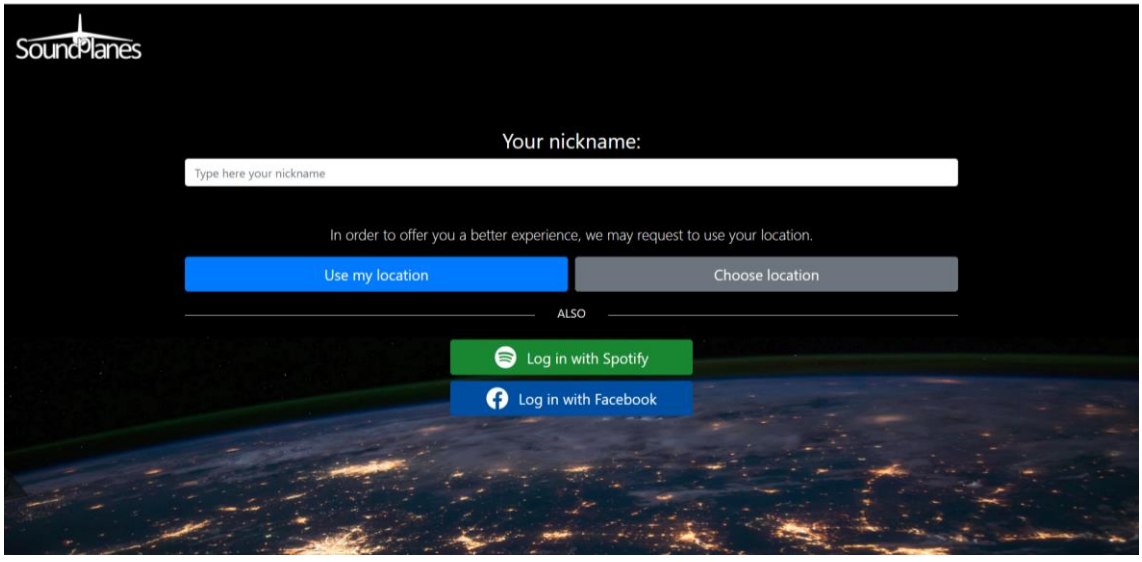
6 User's Manual

6.1 Mashup

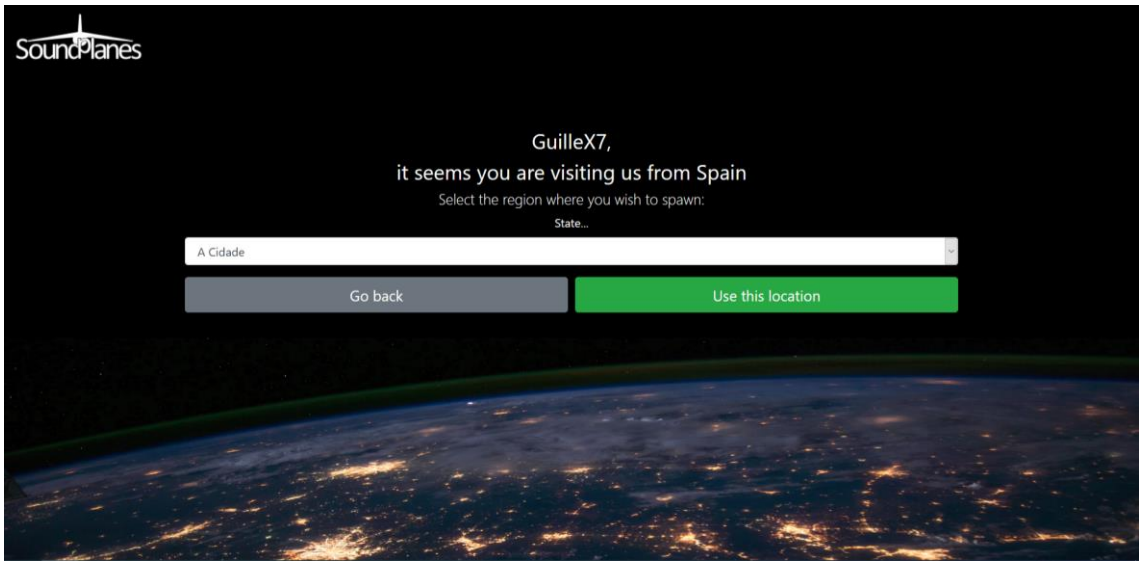
First, landing page will appear on the screen.



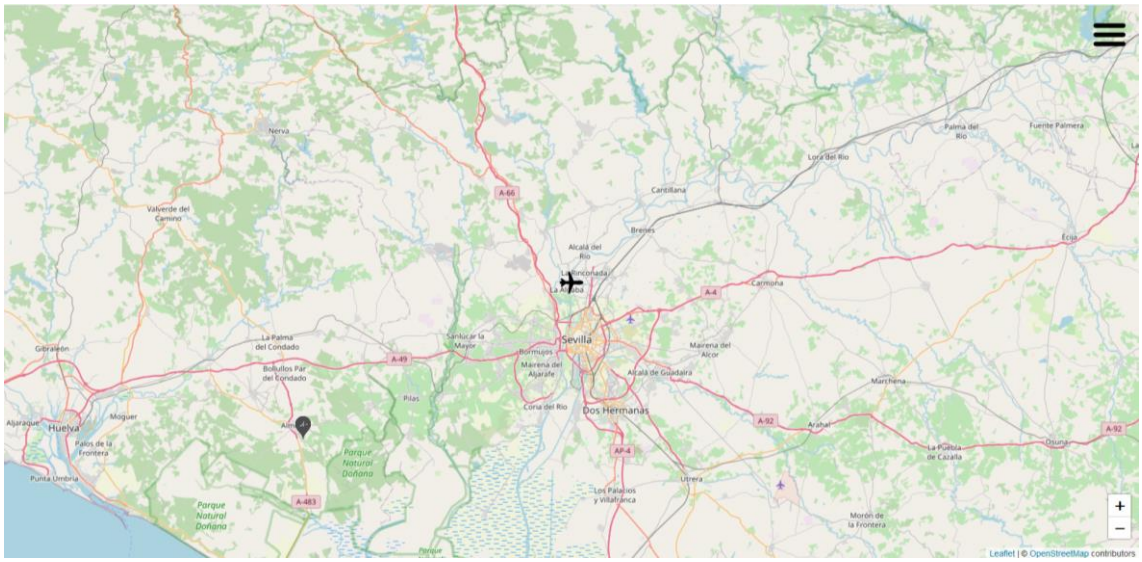
Then, users can log in to the application by writing their nickname and choosing either to geolocate them (by using Geolocation API) or to choose a predefined region coordinates. In addition, users can log in with their social accounts on Spotify and Facebook. If their social account is not linked with any Soundplanes account yet, it will be linked. Otherwise, users will log in directly into the associated Soundplanes account.



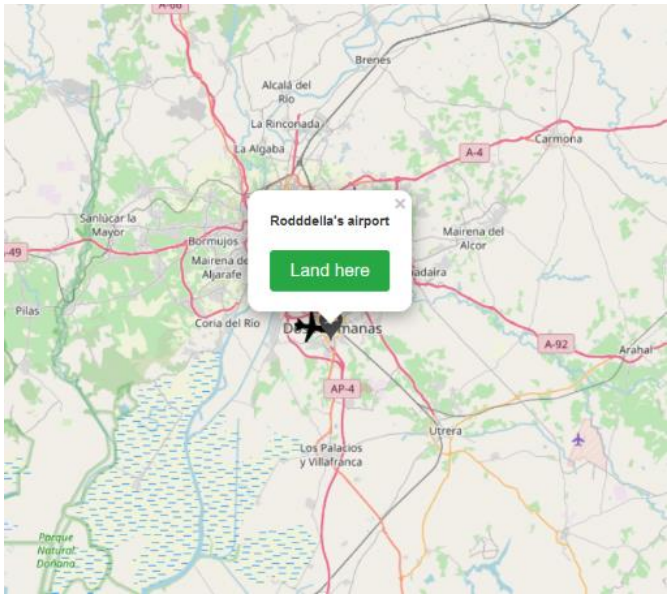
In case the player wants to choose a predefined location, this screen will appear, allowing to choose a region of the country they are supposed to be (determined by browser request headers),

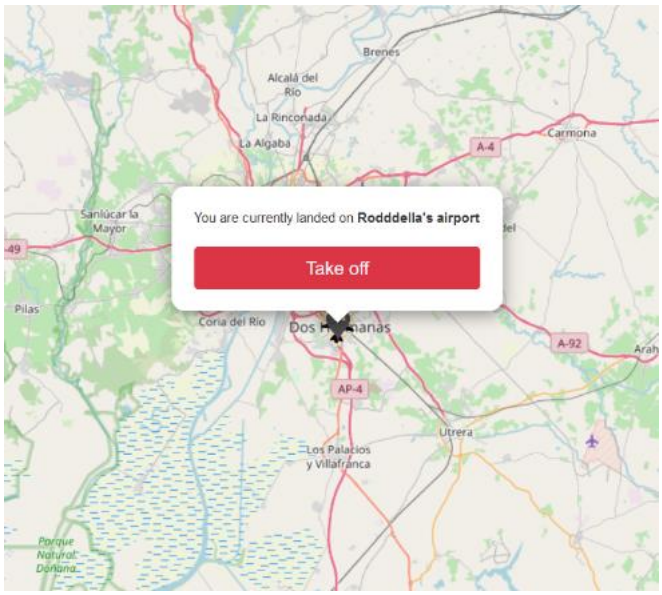


No matter the way the user chooses to log in, he/she will end up on the main application screen:



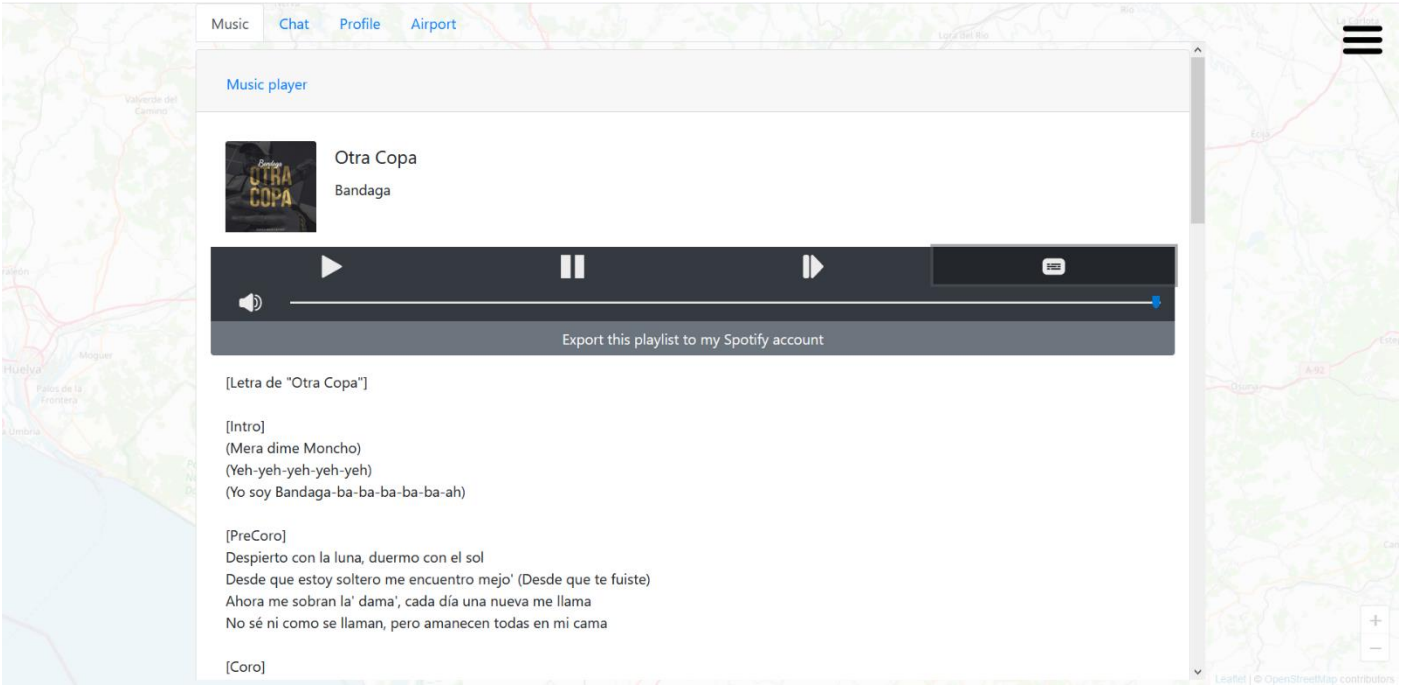
Here, player can travel around the map by clicking/touching the screen. Other users will be represented as flying planes, as well as another user's airports will appear as an airport icon.

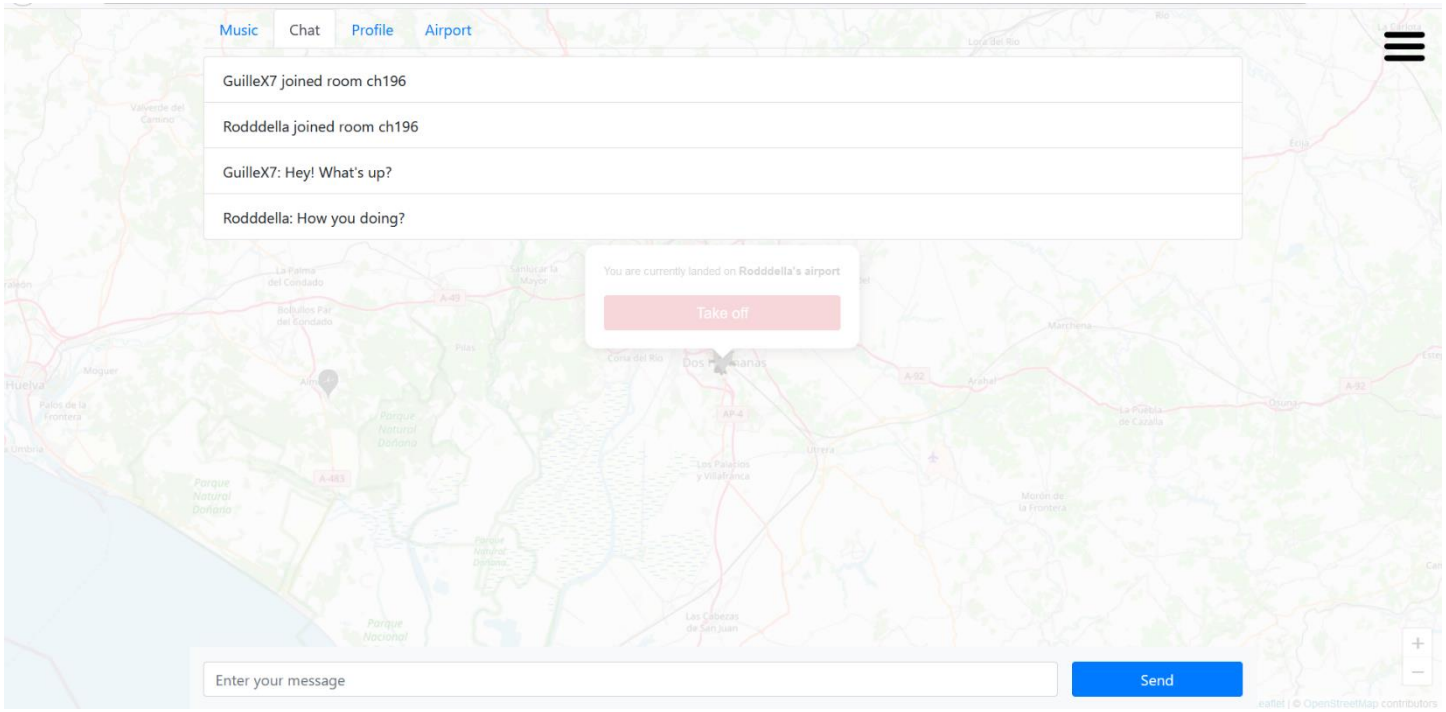




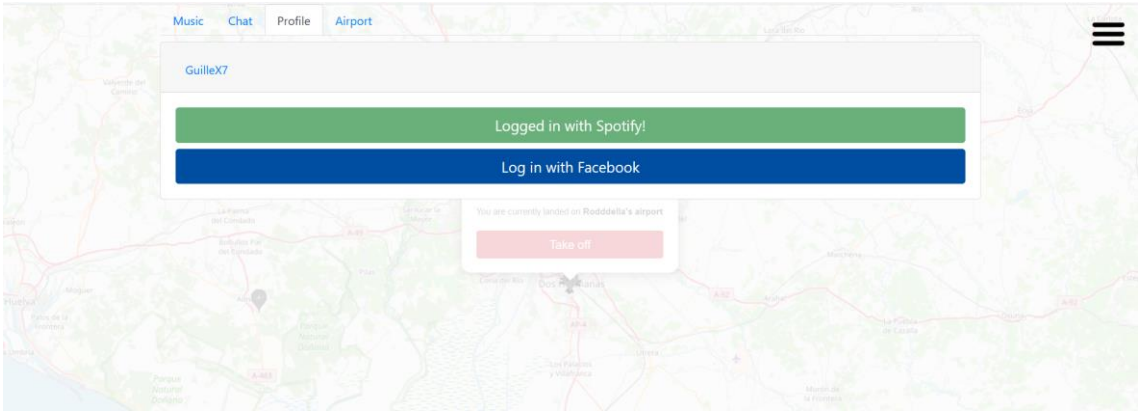
Clicking on an airport will display a popup that allows the user to land on. Landing on an airport will make the user unable to move, but he/she will be able to chat with users that are landed too and listen to the playlist music that is loaded in the airport while reading the track lyrics.

User can click the hamburger button at the top right corner of the screen to toggle the main overlay at any moment. All tabs are always available, but their content will change depending on the user status: whether he/she is logged in with Spotify, if he/she is landed, etc.

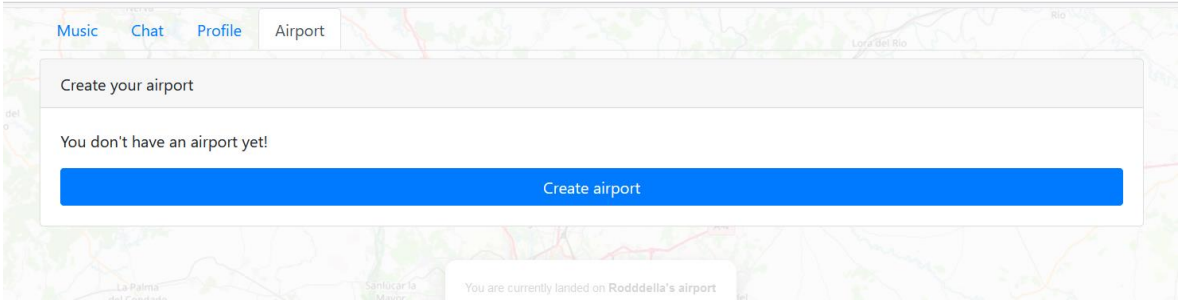




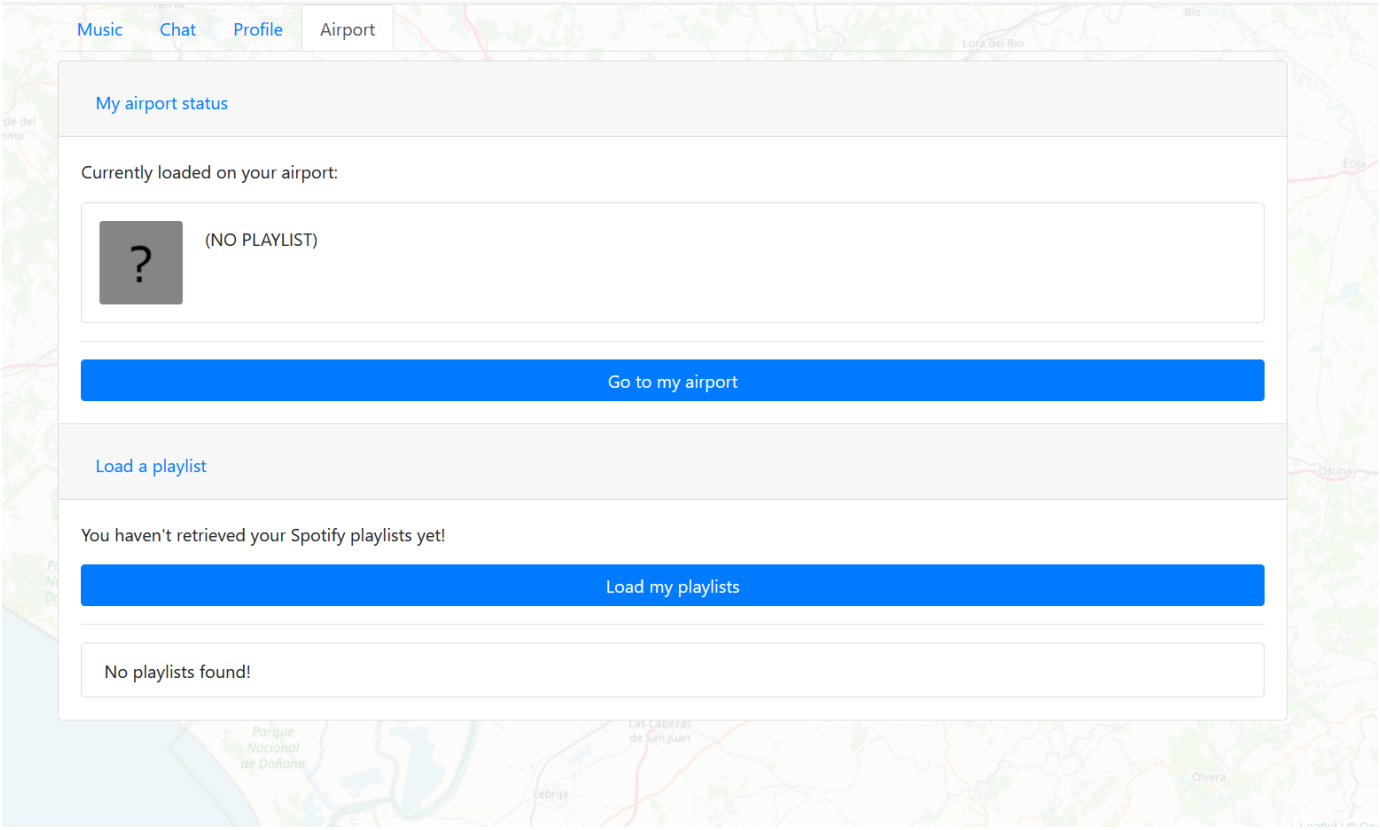
Profile tab shows the user nickname and allows the to log in with social accounts, in case the user didn't before. Otherwise, buttons will be disabled.



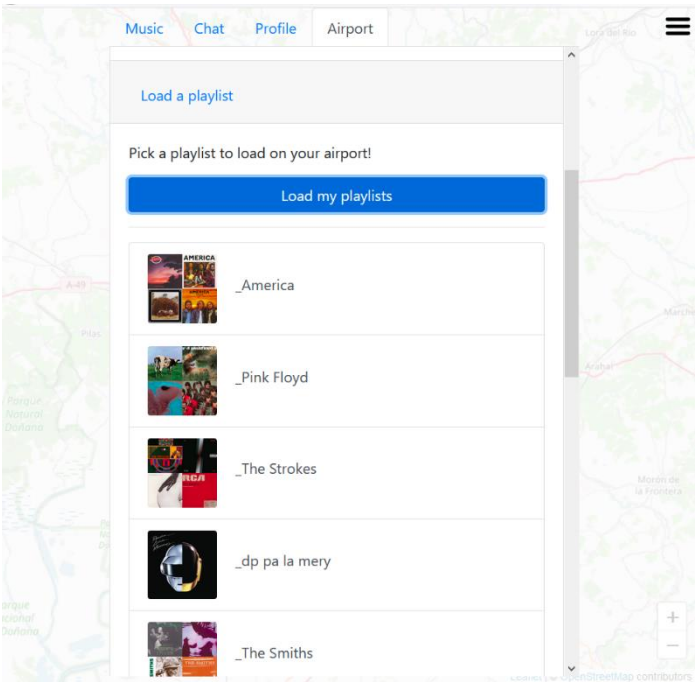
Finally, the airport tab will allow us to create an airport (if we don't have one already) and manage it:

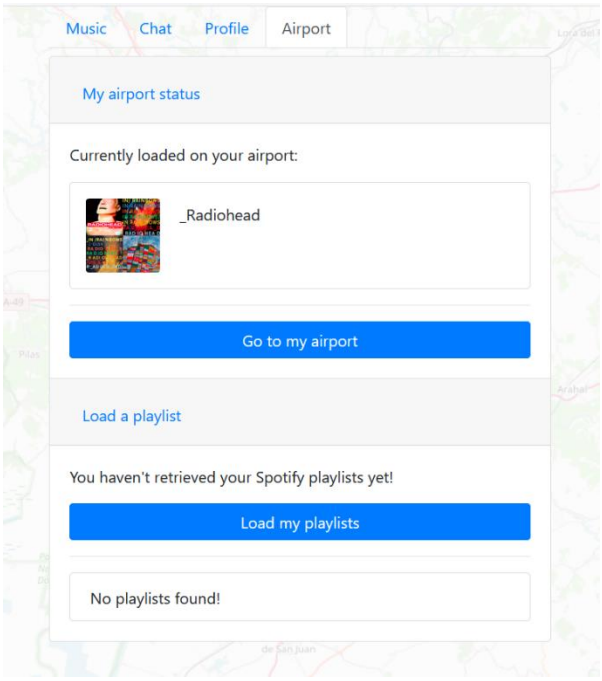


After creating an airport, airport tab will change to something like this:



The current loaded playlist will appear in the top of the tab. Also, a button allows the user to automatically travel to his/her airport. User can load his/her Spotify playlists and load them in the airport.





6.2 API REST

Interactive API documentation can be found at:

https://soundplanes.appspot.com/docs/

or

https://app.swaggerhub.com/apis/Soundplanes/Soundplanes/1.0.0

OAS YAML file can be found at:

https://soundplanes.appspot.com/docs/swagger.yaml

References

[1] *Balsamiq*. http://balsamiq.com/. Accesed on March 2020

[2] J. Webber, S. Parastatidis y I. Robinson. *REST in Practice: Hypermedia and Systems Architecture.* O'Reilly Media. 2010.