Performances Android App

First task - Mobile Computing

Jorge Laguna - up202111636@fe.up.pt

06/18/2022

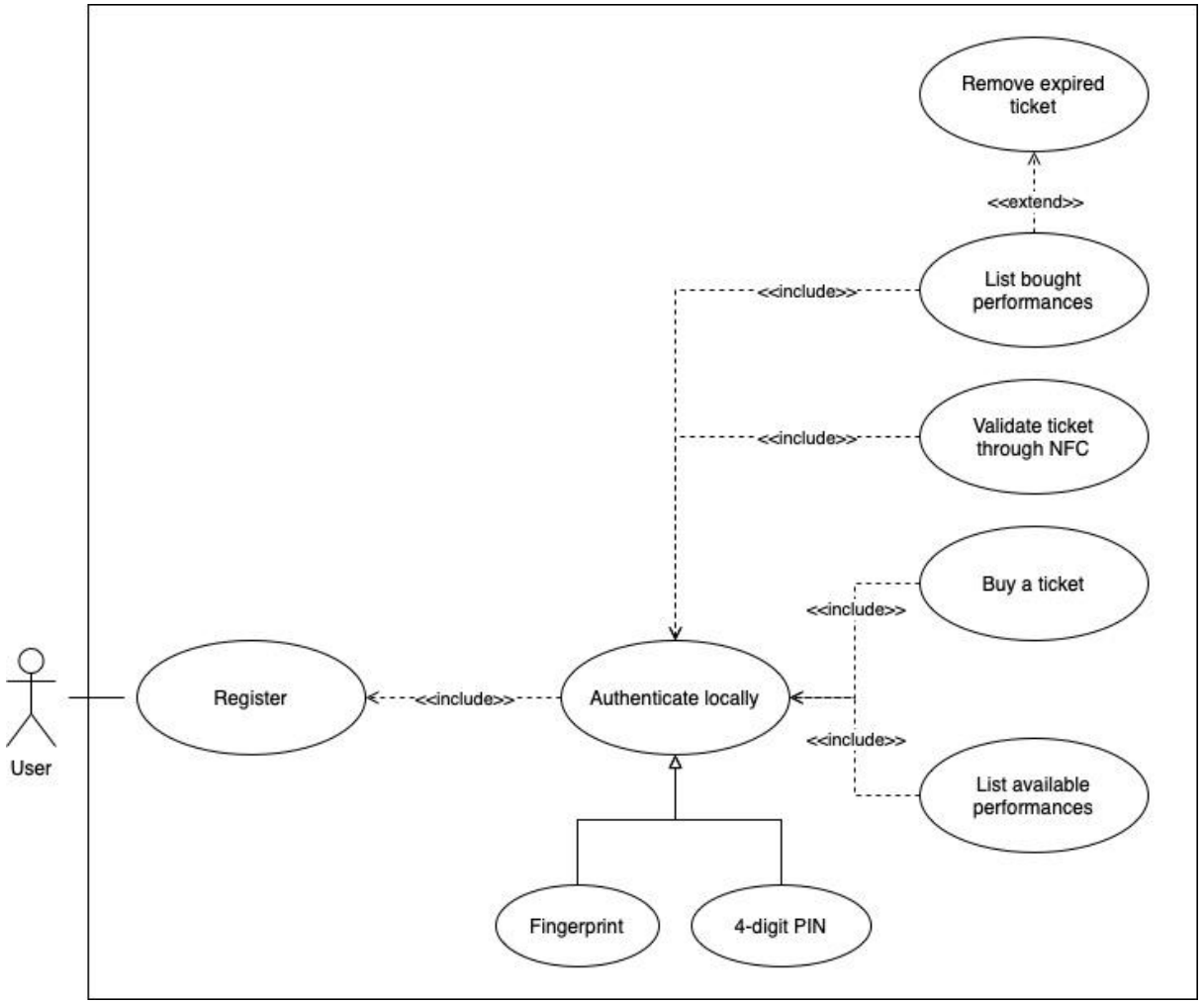
Functional requirements

1. The system should be able to register a new user supplying name, NIF and credit card information
2. The system should be able to show users all the next performances to be presented in the theater (the ones with earlier date will not appear) with info: icon, title, dates, prices and seats left, even the sold out performances
3. The system should be able to let the users buy a given number of tickets
4. The system should be able to let the user authenticate locally via 4-digit PIN or biometrics
5. The system should be able to show a electronic representation of the tickets bought in the app with info: QR, title, date, row-seat number, including the ones which have been expired
6. The system should be able to let the users download their expired bought tickets (not automatically).
7. The system should be able to let the users delete an expired bought ticket from the view.
8. The system should be able to let the users present a set of maximum of 4 tickets through NFC to the validation terminal of the same show with the same performance date
9. The validation terminal should be able to show, in a very visible way, if all the tickets were validated or not.
10. Before the performance begins, the validation terminal should download the sold tickets for the show and verify if valid or not. In either case, the state is changed to ‘used’

Non-functional requirements

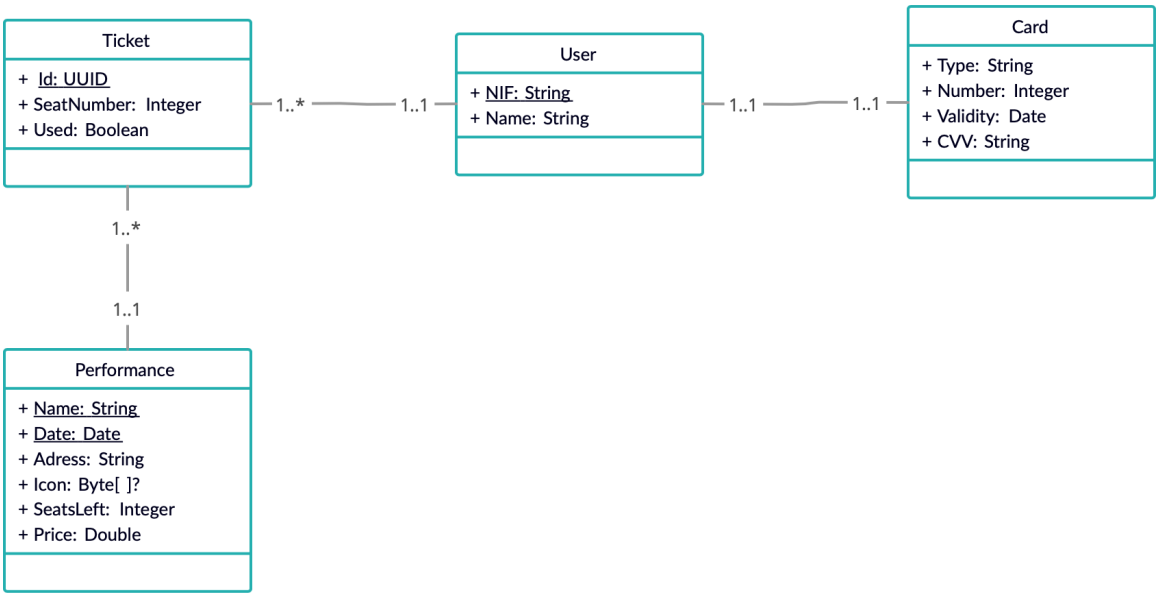
1. The app should generate a cryptographic RSA key pair and transmit it to the server
2. The maximum number of tickets to buy for each user is 5 if available
3. The sold out performances should be visible but grayed out
4. The local auth information should be stored in KeyStore in the device
5. The transmitted info to the validator should contain: user id, number of tickets, tickets ID, and the performance date
6. Bought tickets are stored in an sqlite database in the Android phone so they can be recovered locally without internet connection
7. The expired ticket deletion should be done without Internet connection. Deleted tickets id are saved temporarily in the phone so it can be sent to the server when there is an internet connection

Use cases diagram

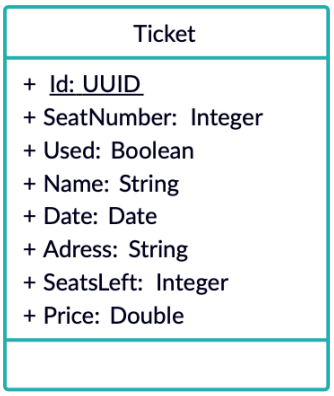


Database design

JPA + Hibernate Database



SQLite



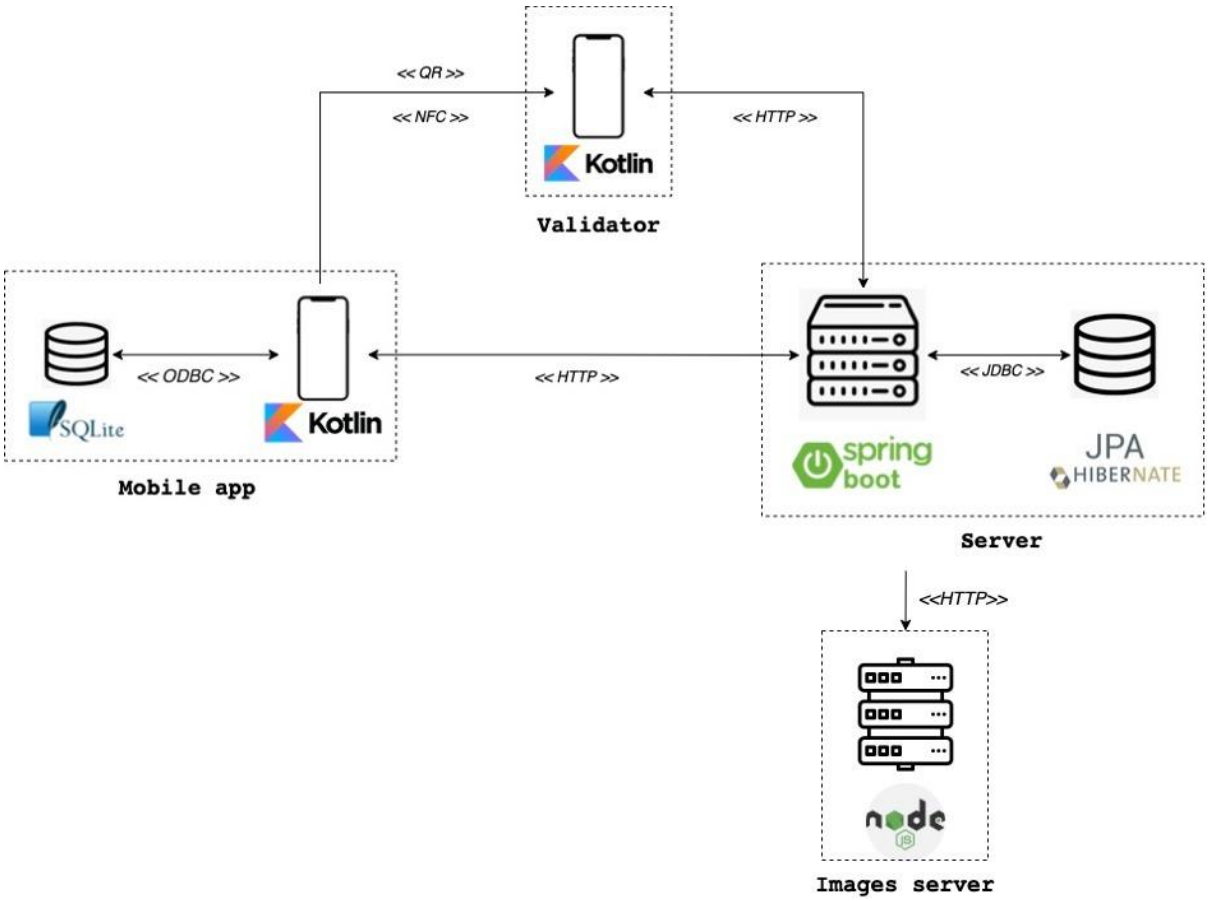
For the moment, we only store the user's bought tickets. The user id is stored in safe KeyStore storage.

RestAPI design

Resource	Method	Explanation	Arguments	Signed with RSA key
/users	POST	Register new user.	- name - nif - card info	☑
/performances	GET	Fetch all performances with their information.	- numPage (pagination)	
	POST	Post new performance.	- name - startDate - endDate - address - price - seatsLeft - imageURI - description	☑
/performances/id	GET	Fetch the description of a performances, which may be too large to load it earlier.		
/tickets	POST	Buy the performance and generate a new ticket for the user. Triggers a query that reduces the seats left for the performance.	- userId - seatsNumber - performanceid - numberBought	☑
/tickets/validate	GET	Returns an error if the ticket is not validated.	- userId - ticketId - performanceId	

System design

Architecture diagram

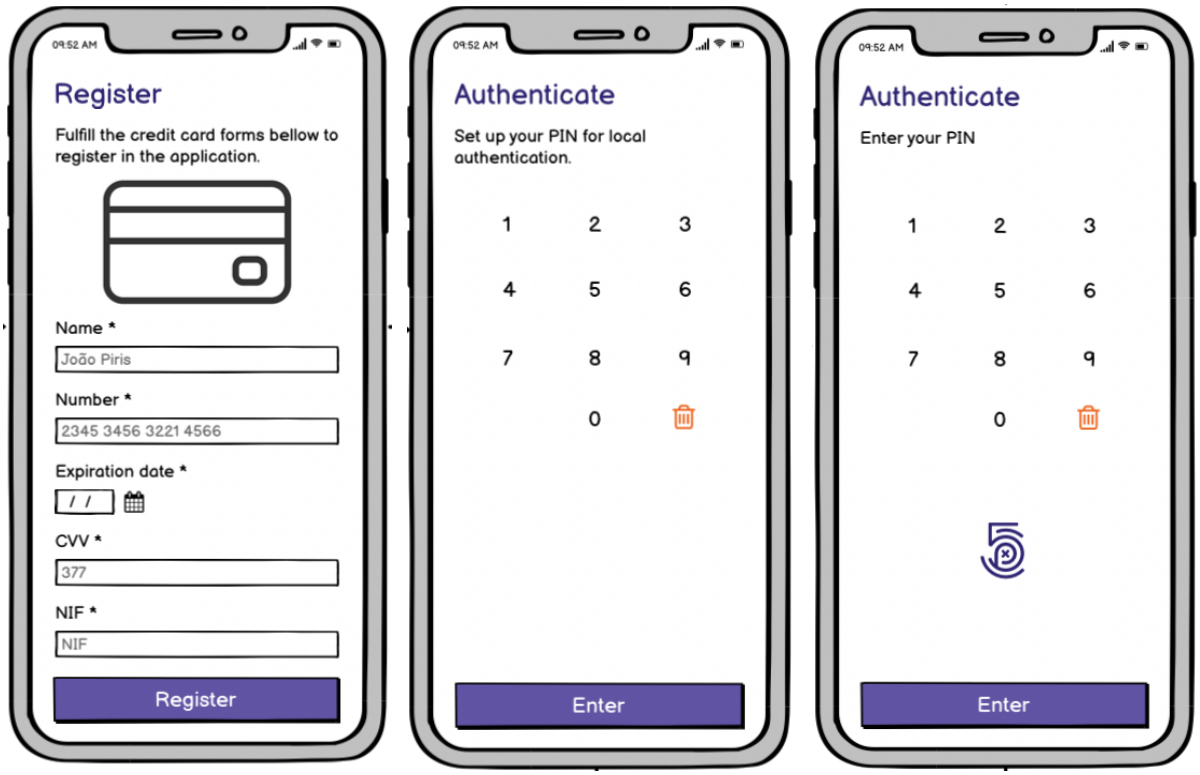


The Spring Boot backend is designed as a hexagonal architecture in order to make the services scalable in time, if the application gets bigger and changes to other technologies. It allows us to save time by using its “magic” annotations and introduce authentication and authorization easily. We will use its local database with JPA and Hibernate as it is a simple database implementation. As we use an hexagonal architecture, we can change our persistence model with ease, keeping the internal app structure.

On the other hand, we will use native code in our Android mobile application with Kotlin. For the local database were we will keep the bought tickets information, we will set up an SQLite database and a KeyStore storage to store both the user id and the public and private key that we generate.

For the validator, we will use another Android mobile phone to emulate the behavior of a real validator, having the posibility to obtain information through QR code and NFC, being able to communicate with our Spring server.

Screens


09:52 AM
Register
Fulfill the credit card forms bellow to register in the application.
Name *
João Piris
Number *
2345 3456 3221 4566
Expiration date *
/ /
CVV *
377
NIF *
NIF
Register
09:52 AM
Authenticate
Set up your PIN for local authentication.
1 2 3
4 5 6
7 8 9
0
Enter
09:52 AM
Authenticate
Enter your PIN
1 2 3
4 5 6
7 8 9
0
Enter


09:52 AM
Performances
search
Cirque Du Soleil
x 20 left
04/20/22 12,50 €
Les Miserables
x 3 left
04/22/22 9,50 €
The lady and the ...
Sold out
04/22/22 10,00 €
The greatest sho ...
x 3 left
04/30/22 15,50 €
09:52 AM
My tickets
Cirque Du Soleil
04/20/22
20:30 - 22:45
Rua da Constituçao 23
Seat number: 23
12,50 €
x 3
Tap to show the QR closer or validate with NFC
09:52 AM
Settings

09:52 AM

Buy tickets

Cirque Du Soleil

04/20/22

20:30 - 22:45

Rua da Constituçao 23

1

Seat number: 5

I declare to have read the Privacy Policy and agree to it.

Total price

12,50 €

Buy tickets

09:52 AM

My tickets

Show past tickets

09:52 AM

My tickets

Cirque Du Soleil

Show this QR or use NFC to validate the ticket

09:52 AM

Past tickets

search

The lady and the ...

x 1

04/22/22 13,00 €

The lady and the ...

x 2

04/26/22 12,50 €

09:52 AM

Past tickets

search

The lady and the ...

x 1

04/22/22 13,00 €

The lady and the ...

x 2

04/26/22 12,50 €

The lady and the ...

x 1

Remove

09:52 AM

Past tickets

The lady and the beas

04/20/22

20:30 - 22:45

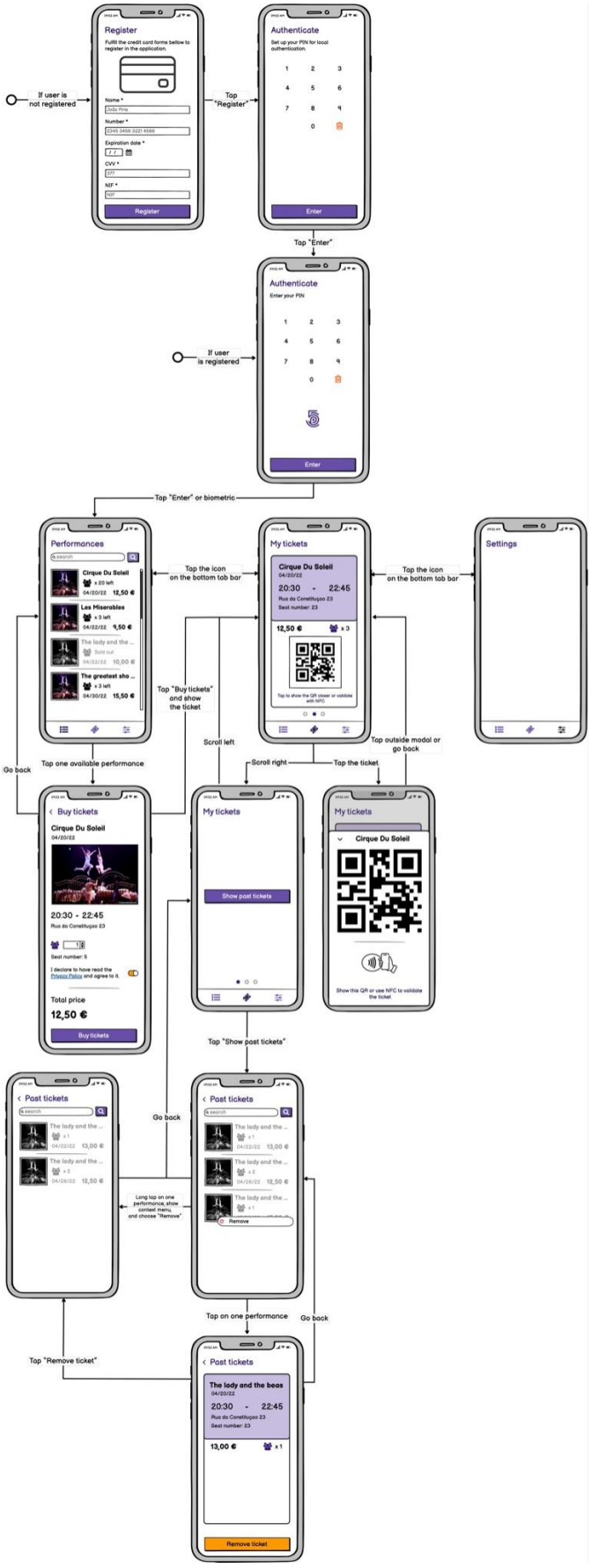
Rua da Constituçao 23

Seat number: 23

13,00 € x 1

Remove ticket

Navigation map



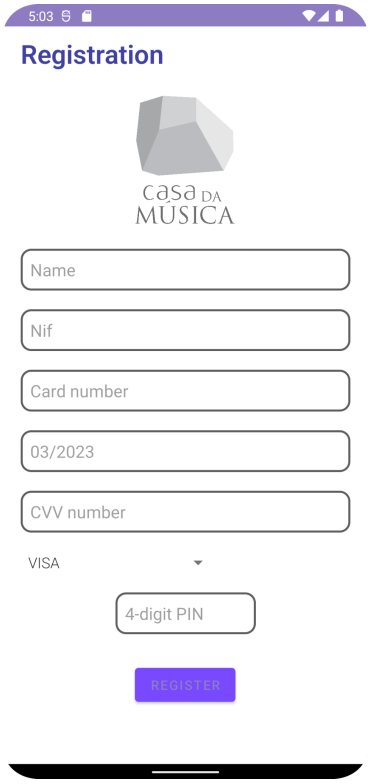
Colors to use:

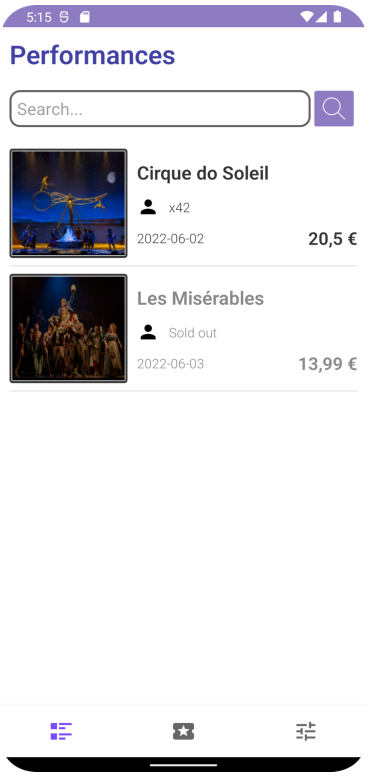
- Primary: Buttons (#674EA7), icons (#351C75)
- Secondary: Buttons (#FF9900), icons (#FC5401)

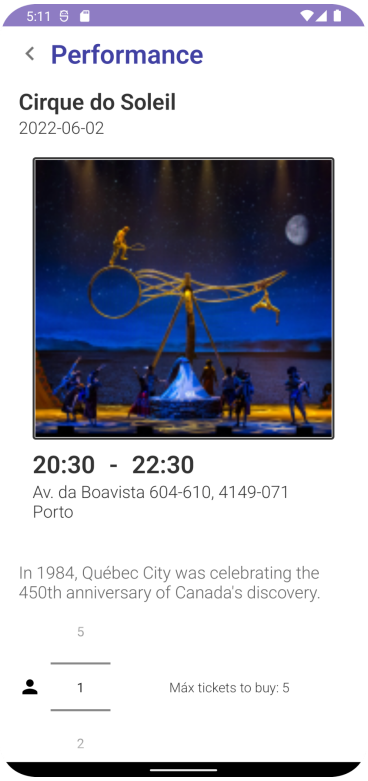
Text style:

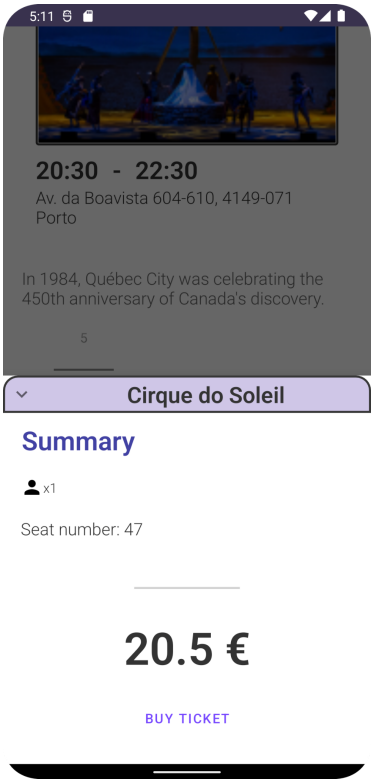
- Font size: page title (32), title (28), text (20), other important text (40)
- Font color: page title (primary icons), title (#000000 - bold), text (#1A1A1A), other important text (#000000 - bold?)

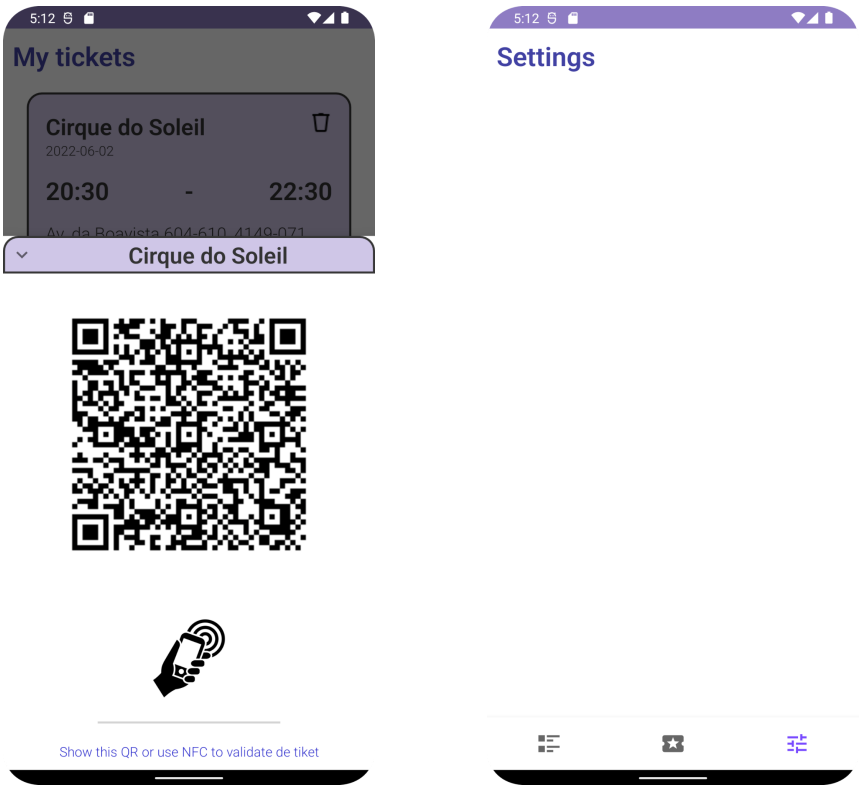
Final screens



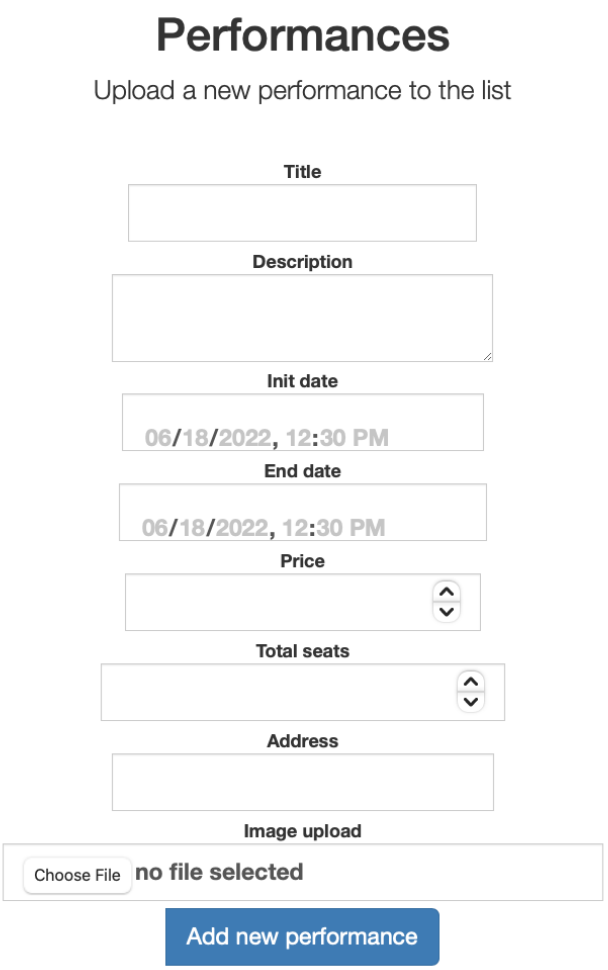








In the end, comparing it to the designed screens, they are pretty similar. The login screen has implemented a simple text input instead of a complete number keyboard to save some time. Moreover, the “Past tickets” screens, which would make an HTTP request to the server to recover other past expired tickets are not finally implemented due to lack of time.



On the other hand, we have a simple form to upload the performance metadata, which is stored in the JPA database and the performance image, which is stored in a NodeJS little files server.

Performed tests

All tests are manually made. They are specified in https://github.com/FEUP-practices/performances-app#testing.

Bugs

All bugs are registered in https://github.com/FEUP-practices/performances-app/blob/main/README.md#bugs.

Notes

Private and public keys are generated on the device. The public key is sent to the server. When it receives it, the server sends an uuid back to the user. When a user wants to buy a ticket, the info has to be signed with his private key, so the server can decrypt it and check if it is the user. When a user consults their tickets the server encrypts it with his public key so the client can decrypt it with the private key avoiding MITM attacks.

We assume that the performances are played in the same city (not specified), but not in the same place.