

POLITECNICO DI TORINO

Department of Electronics and Telecommunication (DET)

Master Degree Program in Engineering

Communication and Computer Networks

Master Degree Thesis

Dynamic frontend design for ticketing and reservation application



Supervisor

Prof. MALNATI GIOVANNI (DAUIN)

Candidate

Niloufar Zahiri

December 2020

Dedication

This study is wholeheartedly dedicated to my beloved father and mother...and my love Roham for all his kindness supports.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Giovanni Malnati for allowing me to do my thesis under his supervision and providing invaluable guidance throughout this thesis.

A special thanks to my friend Ali, my great advisor, for all his pieces of advice and inimitable supports. This never could be done without his help and patience. I would like also to thank Ascanio Orlandini, my technical manager, for all his supports that made this path easy for me to achieve this goal.

LIST OF ACRONYMS

SDLC Software Development Life Cycle

JS JavaScript

AWS Amazon Web Services

API Application Programming Interface

App Application

OS Operating system

IDE Integrated Development Environment

UI user interface

Fb Facebook

Contents

LIST OF ACRONYMS	4
List of figures.....	7
1. Chapter 1.....	8
1.1. Introduction	8
1.2. Targets	10
1.3. Ideas.....	10
1.4. Marketplace	11
1.5. Frontend Marketplace.....	12
2. Chapter 2.....	13
2.1. Introduction: Software life cycle	13
2.2. Propose of SDLC.....	13
2.3. SDLC Cycle.....	14
2.4. Requirement gathering and analysis.....	15
2.5. Design.....	17
2.6. Implementation or coding.....	18
2.7. Testing	18
2.8. Deployment	19
2.9. Maintenance.....	19
3. Chapter 3.....	20
3.1. Tools	20
3.1.1. JavaScript.....	20
3.1.2. XCode.....	22
3.1.2.1. Build and Run the App	25
3.1.3. React Native	26
3.1.4. AWS	27
3.1.5. Bash script	28
3.1.5.1. Switch app	28
3.1.5.2. version app.....	28
3.1.5.3. Factory loader	29
3.1.5.4. Android APK.....	29
3.1.5.5. Run Android	30
3.1.6. PHP.....	30
4. Chapter 4.....	31
4.1. Implementation.....	31

4.2.	Macro Components.....	31
4.2.1.	Platform/server	31
4.2.2.	Frontend merchants / Marketplace	32
4.2.2.1.	Frontend website for Marketplace.....	32
4.2.2.2.	End-user frontend	33
4.2.2.3.	Help Desk support tools:	33
4.3.	App instruction (frontend).....	34
4.3.1.	Home page.....	34
4.3.2.	Around me	37
4.3.3.	Operations.....	40
4.3.4.	Search page.....	49
4.3.5.	Login.....	58
4.3.5.1.	Google Login.....	59
4.3.5.2.	Facebook Login	62
4.4.	Backend APIs	65
4.4.1.	Branch.....	65
4.4.2.	PHP.....	65
4.4.3.	AWS	65
4.4.4.	APIs and RestCalls	66
5.	Chapter 5.....	70
5.1.	Conclusion.....	70
5.2.	Limitation and Future works to improve application	71
5.2.1	Adding live chat assistance:	71
5.2.2	Payment methods:.....	71
5.2.3	Biometric Authentication:	71
5.2.4	Cloud servers:.....	72
5.2.5	Makes app simpler:.....	72
	Bibliography	73

List of figures

Figure 1: SLDC cycle	14
Figure 2: Mypass offer	16
Figure 3: Marketplace launch	17
Figure 4: XCode setting.....	23
Figure 5: Navigation bar.....	24
Figure 6: Device selector.....	25
Figure 7: Switch App.....	28
Figure 8: App version.....	29
Figure 9: Update information	29
Figure 10: Build app	30
Figure 11: Run Android.....	30
Figure 12: Work Fellow	35
Figure 13: Screen navigation.....	36
Figure 14: Home page	37
Figure 15: Geolocation	38
Figure 16: Search page	39
Figure 17 Not found	40
Figure 18: Operations	41
Figure 19: QRCode scanner1	42
Figure 20: QRCode scanner2	42
Figure 21: Ready scanner	43
Figure 22: Manual insertion	44
Figure 23: Successful.....	45
Figure 24: Not valid.....	46
Figure 25: Details	47
Figure 26: Confirmation	48
Figure 27: Search page 1	49
Figure 28: Search page 2	50
Figure 29: Search page 3	51
Figure 30: Search page 4	52
Figure 31: Reservation.....	53
Figure 32: Availability.....	54
Figure 33: Reserve details	55
Figure 34: Payment.....	56
Figure 35: Confirm payment	57
Figure 36: Login screen.....	58
Figure 37: Login status	63
Figure 38: Response	63

1. Chapter 1

1.1. Introduction

The project is in the field of App frontend design for ticketing and reservation application. The application will be developed using React Native platform and build for iOS and Android OS and it features, among other highlights, advanced payment solutions with credit card tokenization and Apple Pay.

Specifically, the App will develop a dynamic graphics frontend for home page and search pages of the application, involving geolocation. Below the specific activities planned.

1. Interface with REST web services to acquire data;
2. Dynamic home page design with real-time resizing of each element according to user screen size;
 - 2.1 Places selection area (ordered by distance from user position among places available obtained from server);
 - 2.2 Category selection area (providing all categories available dynamically obtained from server);
3. Search result pages design with real-time resizing of each element based on user screen size;

3.1 Search by position (around me) and by place selection, with the list of available; tickets/location/product store grouped by category;

3.2 Search by category with a list of available tickets/location/product store grouped by places in order of distance from the user.

3.3 Each store icon displays information of places availability, distance from the user, description, and have specific order based on these values.

This thesis also includes two mockup graphics produced by the UI team of the company as an indicator of expected final work.

We believe this is not just implementation work but there is also some degree higher-level coding techniques to make the user experience fluent and engaging for the purchase experience, such as a smart way to load graphical icons progressively following user's browsing experience without requiring storing all graphics in the app which will make it heavy while at the same time not stressing the network connection that will make the application overall slow.

Aspects of dynamically sized graphics elements based on screen size, handling real-time data from a centralized server and peculiarities of the different operating systems (iOS and Android) combined with the hybrid framework of React Native will be a reasonable challenge.

1.2. Targets

- Rapid implementation to respond to the imminent question raised by the emergency COVID-19.
- Oriented to solve simple problems simplistically.
- Create new assets which integrate the long-term MyPass offer.
- Reuse the tools already built as much as possible.

1.3. Ideas

MyPass believes that it can meet the demands of managing queues and conveniently accessing limited resources for both end-users and merchants, providing a modern B2B booking app, easy to use for merchants and easy to use for end-users.

There is no question that many other businesses have understood this desire, from sites that already deal with reservations and interactive files (like youfirst) to other concepts that are adapting themselves to exist in these deep and stormy seas.

MyPass is motivated to undertake this activity because:

- Some of the necessary components and skills have already been developed and can be re-used.
- The new components to be developed form part of the MyPass Roadmap and the Overview and strengthen other products and services already provided on the Ski and Tourism Platforms (Venice).

- it is an opportunity to activate a new, highly scalable B2B channel that will, again, in this case, complement the existing B2C activities on the Ski and Tourism Platforms (Venice).
- is an incentive to develop and produce the Marketplace System intended for merchants.
- provides us for the scalability and reproduction of the Venice Project in other tourist cities such as Florence.

1.4. Marketplace

The term "MyPass Marketplace" refers to a B2B platform that allows merchants to register independently and become part of the MyPass offer on different future operating platforms, currently Ski and Venice, and probably the booking portal. For example, a ski resort restaurant could register to receive payments with the MyPass Ski Card, while a tour operator could provide concert tickets in Venice, another bathhouse operator could book/sell "umbrellas" regularly or a caterer to handle reservations and waiting lists.

Registration should be easy and can be done separately and must require a collection of text and graphic details to be compiled individually.

Subject to content verification by the MyPass operator, live publication by the merchant must be almost automatic for several predefined operating modes.

1.5. Frontend Marketplace

The Marketplace Frontend must be fast, intuitive and simple.

The most important point is the geolocation system (and the respective map), the possibility of identifying activities or services through well-defined filters both by product sector (e.g. transport, culture), and by place (city, region, state), and budget (e.g. price ranges), both of services offered (e.g. suitable for families, suitable for people with mobility difficulties), and for other needs (e.g. immediate confirmation, cancellation policy).

Filters make it possible to respond promptly to the requests of the final consumer.

The Marketplace home will focus on the most requested or popular cities, businesses or services. An in-depth study will be carried out by the Marketing and Sales department to guide the consumer to purchase.

It should be clear what the service includes and what does not, such as general conditions and important information.

2. Chapter 2

2.1. Introduction: Software life cycle

SDLC is a cycle which characterizes the different stages engaged with the advancement of programming for conveying a top-notch item. SDLC stages spread the total life pattern of a product for example from origin to retirement of the item.

Clinging to the SDLC cycle prompts the improvement of the product in an orderly and taught way.¹

2.2. Propose of SDLC

The inspiration behind SDLC is to produce an outstanding item that meets the customer's requirements.

SDLC has described its phases as the selection, design, coding, testing, and maintenance of specifications. The steps to consistently manufacture the commodity should be adhered to.

Examples involve the development of software and the division of a team to work on a specification of the product and the work they are allowed to do. One developer decides to design first, while the other

decides to code the documentation component first and the other component.

This can lead to project failure, which allows the team members to have a reasonable level of knowledge and insight².

2.3. SDLC Cycle

SDLC cycle is a software development process as it is shown below.

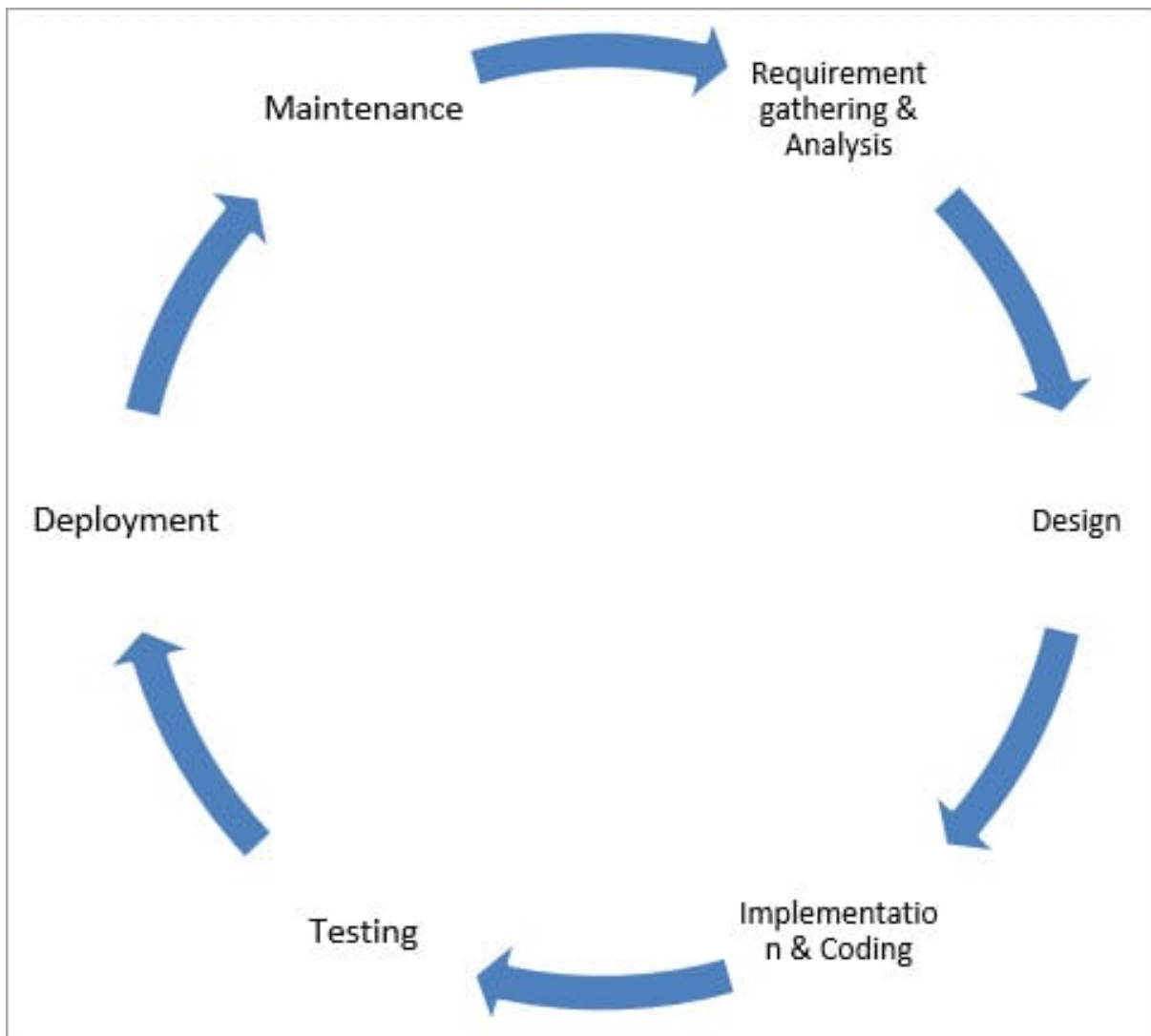


Figure 1: SDLC cycle³

2.4. Requirement gathering and analysis

During this phase, business needs are collected. This step is the project managers 'and stakeholders' main priority. Meetings are being conducted with administrators, stakeholders and users to decide the specifications, such as; who will use the system? How are they going to use the system? What data are to be inserted into the system?

What input should the process output? There are many general questions which are addressed during the meeting process of requirements. These criteria will be analyzed for their relevance after the specifications have been obtained and the possibility of integrating them into the production framework is also explored.

Finally, a design specification document is created to guide the next step of the plan. After the review of the criteria, the development team follows the software test life cycle and begins the test planning.⁴

One Global App for MyPass offer

New App Release - June 30

- Designed as frontend for new Marketplace platform, handling reservations and sales
- Built on one codebase, to easily produce and maintain several sister apps , with possibility to control which events/products/merchants are delivered on each app - Freedom to choose how to deliver content
- Classification and searches by categories and places, finding possibility around the users ordered by distance
- Reservation and Availability centric, meeting COVID-19 requirements and needs



Confidential information property of MyPass Company, not to be further shared without MyPass approval.

M PASS

Figure 2: Mypass offer

Marketplace Launch - July 1

- Interactive frontend for merchants exporting services or access tickets on MyPass platform
- Allows onboarding from different areas, very easily and quickly. New app frontend for Marketplace allows to purchase and reserve services and tickets “around you” and selective places
- Combines the reservation and payment aspects of transactions, designed to address the specific COVID-19 post lock-down needs of merchants requiring to reserve access to limited resources (museums, restaurants, stores, sport activities, beaches, events)



Figure 3: Marketplace launch

2.5. Design

The application design, network design, databases, user interfaces or system interfaces are included.

Transform the SRS text into a logical framework containing the extensive and thorough collection of programming language specifications.

Establish an emergency, preparation, maintenance and organizational plan.

Check the concept suggested. Ensure that the final design must satisfy the SRS document specifications.

Finally, write a specification report for use in the following processes.⁵

2.6. Implementation or coding

The project team generates the final product during implementation. The development of the product can be an exciting process for customers, as their concept is something concrete for the project. Software creation and coding begins with project developers.

If a customer, for instance, needs a new software application, the project developers need to design the application to satisfy the product requirements of the customer. The team must meet unique coding criteria when designing the code. Customer needs can require complex computer programs or enhancements and developers must execute the applications to be able to operate correctly.⁶

2.7. Testing

You are focused on study and exploration at the test phase of the SDLC software development cycle. During the testing process, developers will decide if their code and programming are customer-specific. There was a mistake. The project team creates a test plan before construction can proceed.⁷

2.8. Deployment

The aim at this point is to deploy the program in the production environment to enable users to start using the product. However, many companies, such as a research or a stage setting, prefer to transfer the product across various deployed environments.

This makes it possible for any stakeholder to play with the product safely before it is published. Furthermore, before the product is released, it allows any final errors to be identified.⁸

2.9. Maintenance

The start of the SDLC maintenance process marks as soon as the software product is released. This involves post-production procedures such as system changes and modifications. In this section, the team may also begin preparing the future software-added functionalities and features.

Furthermore, if a problem occurs that needs to be resolved, developers need to be told to make corrections according to the seriousness of the problem. A hot-fix might be required for the product, meaning to repair high-priority features which should be performed as quickly as possible.⁹

3. Chapter 3

3.1. Tools

3.1.1. JavaScript

In our app, we use Js in the react-native framework for the frontend part. Initially, JavaScript was built to "live web pages."

The programs are called scripts in this language. You can enter it right in the HTML of a web page and run as the page loads automatically.

As plain text, the scripts are supplied and executed. No special planning or compilation is required to run.

JavaScript is very different in this regard from another Java language. JavaScript nowadays runs on any computer that has a special program named the JavaScript engine, not just in a browser, but also on its server.

The "simple" programming language is Modern Java-Script. It does not have low-level memory or CPU access since it was originally built for browsers that do not need it.

The capabilities of JavaScript depend heavily on the context in which it operates. Node.js supports functionality for reading / writing arbitrary files, network queries etc.

for example:

JavaScript in the browser will do all about the handling, the user and the webserver of the website.

JavaScript can:

- Add new HTML to the tab, change the text, change styles.

- Click on a mouse, respond to user actions, shift pointers and press keys.
- Requests to remote servers via the network, download and upload files (so-called technology AJAX and COMET)
- Get cookies, ask visitors questions, show messages.
- Remember customer-side data ("local storage").

To protect the user, JavaScript's browser capabilities are limited. The goal is to avoid the access of private information to an unethical web site or to harm the user's information.

Examples of Js restrictions:

- On a web page, Js cannot read/write arbitrary files, copy them or run programs on a hard disk. It doesn't have direct OS access. Modern browsers allow files to be used, but there is restricted access and only accessible if the user performs certain actions, such as "dropping" a file in the browser window and/or selecting it via a `<input >` tag.
- Various tabs/windows do not normally know each other. Often, for instance, if one window uses JavaScript to open the other window. But even if they are from different pages (from a different region, protocol, or port), JavaScript on one page cannot access the other.

- JavaScript will connect to the server from which the current page was originated through the network. However, it is limited in its ability to collect data from others. If possible, explicit (expressed in HTTP headers) consent from the remote side is required. Again, this is a restriction of protection¹⁰.

3.1.2. XCode

our app supports two OS, android and IOS, for the IOS part we will use XCode in order to build the IOS projects, do the TF and debugging for IOS part, the more details will be explained in the next following part. XCode is the Integrated Development Environment (IDE) of Apple. XCode allows you to build apps, including iPad, iPhone, Apple Watch, Apple TV and Mac, for Apple product. XCode offers tools to handle the whole process of your production – from formation to testing, optimization and submission to the App Store.

A project manages the files and resources required to build your app. Start with one of the templates to create a project, and change it as you want. For each platform (iOS, watchOS, tvOS and macOS) as well as for popular application, frameworks and library forms, templates are available. Both templates have default settings pre-configured and are ready to create and execute. To display an interactive preview, pick Swift as the language of programming, and SwiftUI as the user interface when setting the interface and editing code.

The main window appears after you build a project. This window is your primary interface to access, edit and control all your project

components. It is scalable and configurable and adapts to the needs of the task and allows you to customize it according to your style of work.¹¹

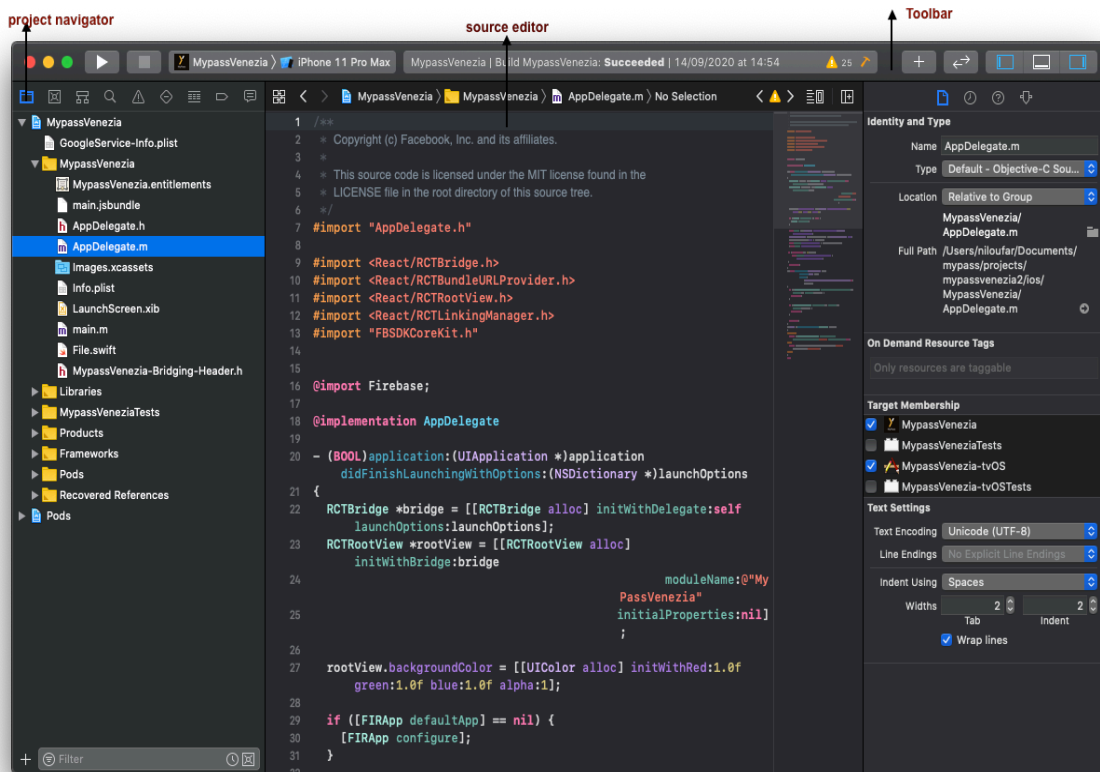


Figure 4: Xcode setting

To quickly access various parts of your project, use the navigator field). To view the appropriate pieces in the content section below the navigator bar, press a button in the navigator bar.

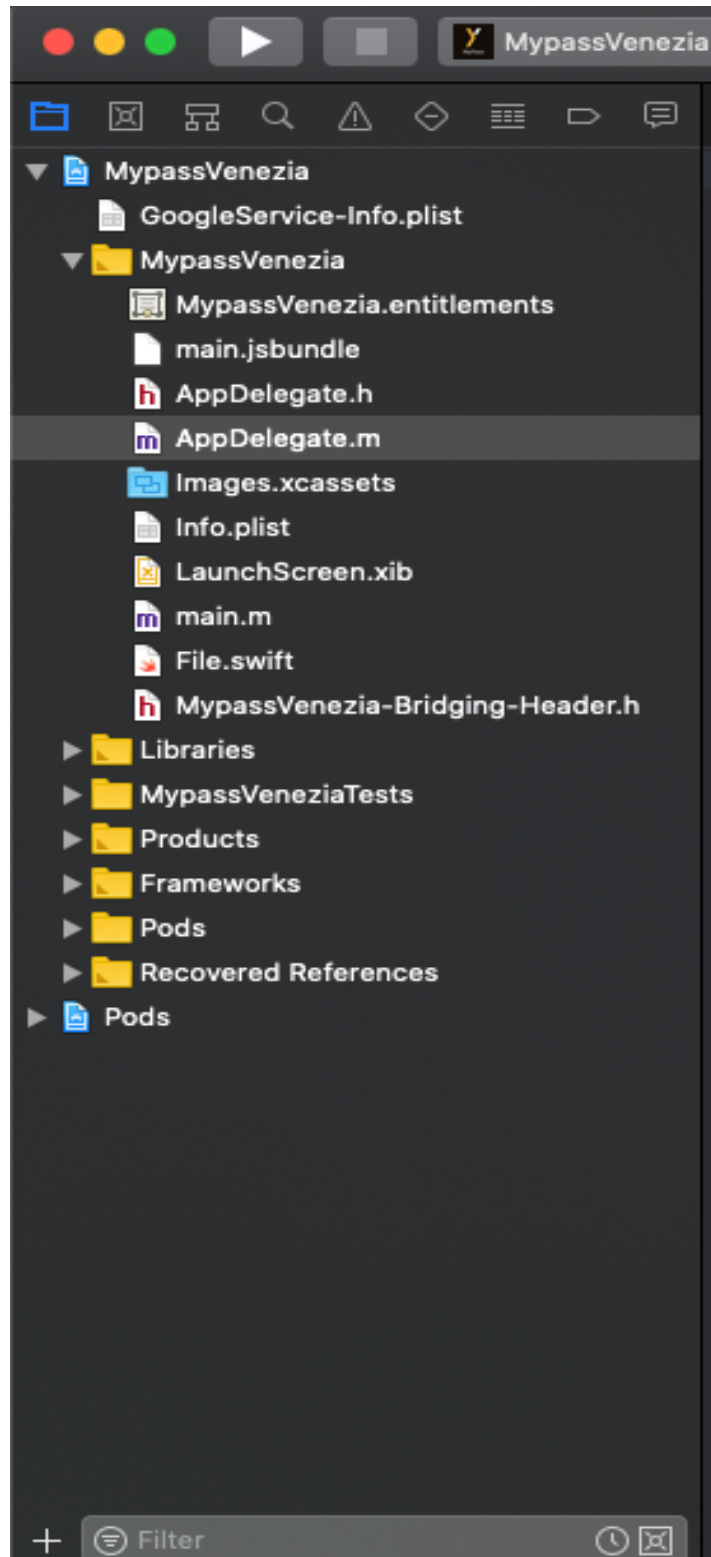


Figure 5: navigation bar

3.1.2.1. Build and Run the App

It is possible to run the app on the simulator or the real device.

The debugging part will be open automatically, as far as the app runs on XCode.

In the following steps, I will explain how to run the app step by step:

- Select a schema in the toolbar from the schema popup menu
- Choose a simulator from the menu of devices



Figure 6: Device selector

- The next step is to click on the run button in order to build, run and debug the app.
in case of a successful build, the app launches and the debugging part opens in the debug area.
- if any error or warning happens, we can check them in the activity area, it shows us the line of error.

3.1.3. React Native

react-native is a framework which combines the best part of native programming language with react, we will use the Js and its library inside the react-native framework for building the user interfaces.

The export of building an app with react native is one app with two different OS, android and IOS.

React Native helps you to create fully native apps which do not affect the experience of your users. It offers a core set of native platform components, such as view, text and image which map the platform's native user interface components directly.

All the components of react-native wrap the native code and interconnect with APIs through React's declarative UI pattern and JS.

Fast Refresh: while we change some part of code, as soon as saving the change, we can see our changes, no need more time to builds only can save, see and repeat.¹²

3.1.4. AWS

Amazon Web Services (AWS) is a branch of Amazon that offers on-demand cloud computing systems and APIs on a robust pay-as-you-go basis to individuals, businesses and governments. These cloud computing web services provide a selection of simple abstract technical infrastructures and computing frameworks and tools distributed. Amazon Elastic Computing Cloud (EC2), one of them, enables users to access a virtual machine cluster that is accessible on the internet all the time. The Virtual Machine version of AWS emulates most of the characteristics of a real computer, including processing central hardware (CPUs) and graphics processing units (GPUs), local / RAM, hard-disk / SSD storage, operating system range, networking, and preloaded applications such as web servers, databases, and client relationship management (CRM).

In our application, our backend operation is done in two separated part based on the cloud on AWS and also the other part was programmed with PHP.

The cloud part on AWS was coding with node.js and also python and JS, we used amazon lambda for monitoring and controlling our app process which programmed with python.¹³

3.1.5. Bash script

In MyPass reservation app during the process, for building the app, increasing the version of the app, switch from test to production, build the APK for android we use bash scripts, which I indicate these parts:

3.1.5.1. Switch app

Our app has two parts, test version and also production one.

The test part is for the internal user to test the application performance and if there are some errors also for debugging, the production one is the finalized version which the company tester approves that all parts of the application work well.

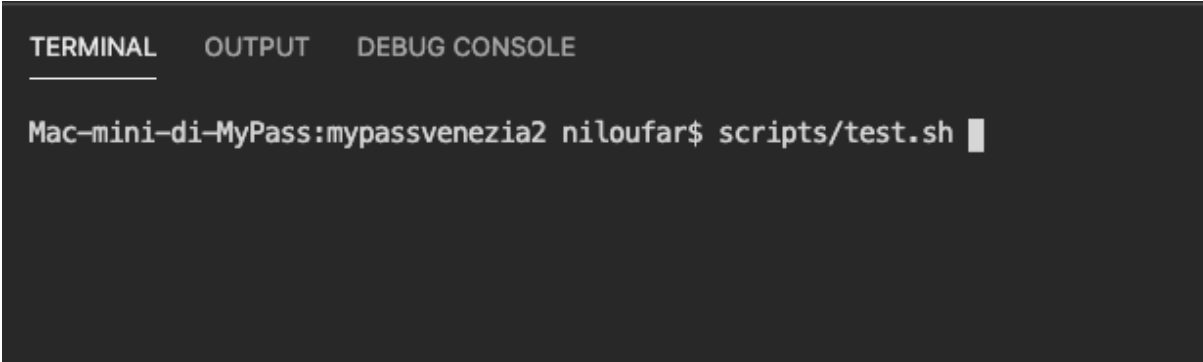
A screenshot of a terminal window with a dark background. At the top, there are three tabs: 'TERMINAL', 'OUTPUT', and 'DEBUG CONSOLE'. The 'TERMINAL' tab is selected and underlined. Below the tabs, the terminal shows the prompt 'Mac-mini-di-MyPass:mypassvenezia2 niloufar\$' followed by the command 'scripts/test.sh' and a cursor. The rest of the terminal area is empty.

Figure 7: switch App

3.1.5.2. version app

each time we debugged, fixed some errors or we add some new features to our app we will increase the app version, which is not manually and we do this in bash script.

```
TERMINAL  OUTPUT  DEBUG CONSOLE
Mac-mini-di-MyPass:mypassvenezia2 niloufar$ scripts/version.py
```

Figure 8: App version

3.1.5.3. Factory loader

when the app is switched to test or production mode, the data which are retrieved from the server is different in test and production, so each time the app is switched we should use factory loader to update the correct information according to the test or production.

```
TERMINAL  OUTPUT  DEBUG CONSOLE
Mac-mini-di-MyPass:mypassvenezia2 niloufar$ scripts/factory_loader.py
```

Figure 9: update information

3.1.5.4. Android APK

since the application is completed, we should build iOS and android, I did a script which automated build the apk for android, to run that script I use the below code.

```
TERMINAL  OUTPUT  DEBUG CONSOLE
Mac-mini-di-MyPass:mypassvenezia2 niloufar$ scripts/build.sh
```

Figure 10: build app

3.1.5.5. Run Android

In the first step, if we want to run the app on the Android simulator or android device to understand how the app works and also to understand the errors, we will use the mentioned code to run scripts on android.

```
TERMINAL  OUTPUT  DEBUG CONSOLE
Mac-mini-di-MyPass:mypassvenezia2 niloufar$ react-native run-android
```

Figure 11: run Android

3.1.6. PHP

All the services which are done in the backend

4. Chapter 4

4.1. Implementation

MyPass provides a series of standard booking models, which can be extended over time according to needs and opportunities. The first implemented would be:

- book a series of services/resources available daily or at certain times (umbrellas, rooms, tennis courts, tables, bicycles, ...)
- manned waiting lists (catering model)
- unattended waiting lists (food counter model)

To discourage empty bookings and for the convenience of the operator, a booking "fee" of the value set by the user and predefined cancellation policies may be required. These amounts can be equal to the price of the purchased product (e.g. cost of the day for umbrella), a deposit (e.g. 10 € discounted from the dinner price) or even just a deposit (e.g. 5 € returned if the user arrives within a certain time).

4.2. Macro Components

The main components of the booking platform are listed below:

4.2.1. Platform/server

- Authentication, registration, login.
- Payments, credit card tokenization, Apple Pay.

- Ticket sale with credit card booking mechanism and subsequent automatic accounting, access obliteration.
- Seat/resource booking logics, waiting in the queue.
- Support services for the registration of merchants (marketplace).
- Support services for merchant portal.
- Disposition of payments with automatic issuance of bank transfers.
- High reliability/redundancy server architecture.

4.2.2. Frontend merchants / Marketplace

4.2.2.1. Frontend website for Marketplace

- Merchant registration.
- Operator operating parameters configuration:
 - general configuration.
 - loading information and graphics content.
 - entering booking/availability data.
- Status of payments due to MyPass.
- Reporting/account statement.
- Sending tickets and assistance.
- Application for validation / access.
- Application for checking / changing availability.
- API / Web Service to enable integration.

4.2.2.2. End-user frontend

- iOS / Android application infrastructure with login/registration functions, dynamic loading of contents/merchants, ticket wallet, credit card tokenization, Apple Pay, social login, ticket sales
- Application of new graphics to the application
- Application specialization with operational booking functions, identification of bookable points nearby
- Responsive website for information only or (possibly, subsequently) also device.

4.2.2.3. Help Desk support tools:

- User and merchant registration.
- Back office support functions for merchants.
- Back office support functions for end-users.
- Support functions for MyPass operators (back office).
- B2B CRM functions for quick management of contacts, contracts, deadlines.

4.3. App instruction (frontend)

4.3.1. Home page

The home page is entire dynamic, according to the user choice like the destination or events all the parameters such as images, text and all the explanations come from the server, each time a service is called to retrieve all the information, from the frontend point of view all parts are made as a component so they are completely responsive.

the home page composed of:

- Destinations which includes about 20 cities in Italy.
- Events and activities which includes restaurants, concerts, beaches, etc.
- Operations which are used for ticket obliteration, control and verify tickets. only the privileged users can have access to this part.

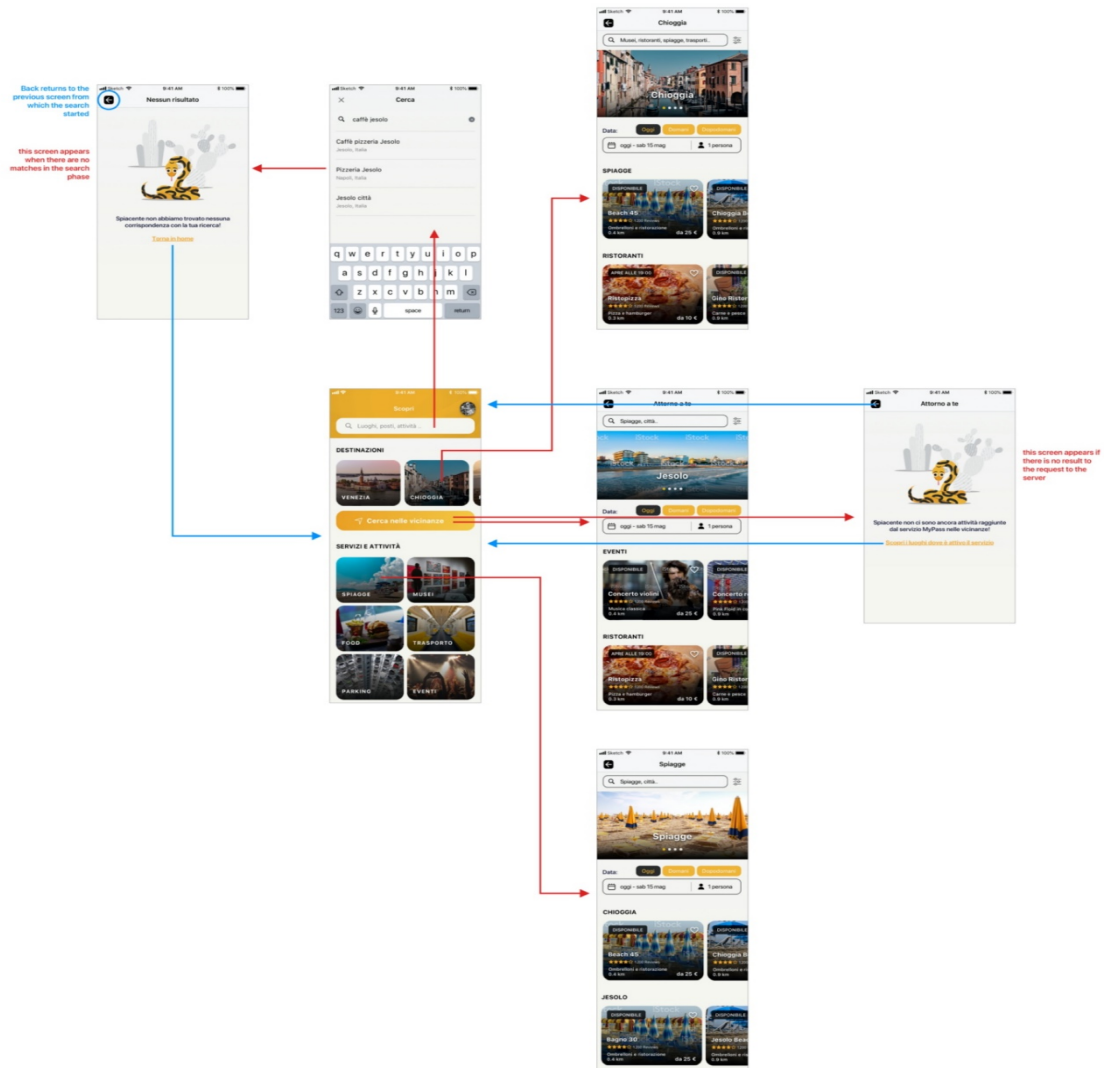


Figure 12: work Follow

The home page is designed according to the user requirements, the user can search based on destination or events or all the places around him. All the contents of the home page are retrieved through a call to the server so the pieces of information are dynamic and whenever the company decides to add or remove some events or cities or change the contents it is very easy to do just by adding or removing the JSON pieces of information.

In our application we use lots of libraries, and also, we need to use state and props in react-native to update each time our information according to the part that we are.

As demonstrated in the below picture, components are built and then all the parameters passed to them as props or states so all the Information can be navigated through different screens.

```
render() {
  // console.log('state is', this.state);
  // console.log('[MyPassHome json]', this.props.json);
  const isInfoAvailable = this.props.json && this.props.json.HOME_CONTENT;
  return (
    <View style={styles.maincontainer}>
      <Header headerOffset={this.state.headerOffset}
        goProfile={this.props.goProfile}
        onSearch={this.props.onSearch}
        onPress={this.props.onPress}
        onSearchSubmitted={this.props.onSearchSubmitted}
        numTickets={this.props.numTickets}
        logIn={this.state.logIn}
      />

      <ScrollView style={styles.body}
        scrollEventThrottle={1}
        onScroll={{(e) => { this.getHeaderHeightPortion(e.nativeEvent.contentOffset.y) }}>
        <View style={{ margin: 5 }} />
        <View style={styles.headerTitles}>
          <Text style={styles.destinationTitle}>{_("Destinazioni").toUpperCase()}</Text>
        </View>
        <View style={{ margin: 5 }} />
      </ScrollView>
    </View>
  );
}
```

Figure 13: screen navigation



Figure 14: Home page

4.3.2. Around me

MyPass reserve app has a feature that the user can find all the closest events, restaurants, concerts around him.

Thanks to the “Geolocation” API which helps the developer to get the user coordinate and then shows all the nearest events.

The library should be imported as:

```
import Geolocation from '@react-native-community/geolocation';
```

```
getUserLocation = () => {
  Geolocation.getCurrentPosition(
    position => {
      console.log('position', position);
      this.setState({ lat: position.coords.latitude, lon: position.coords.longitude });
    },
    error => {
      console.log('error', error);
    },
    { enableHighAccuracy: false, timeout: 20000, maximumAge: 1000 }
  );
}
```

Figure 15:geolocation

Then this function will be called to get user coordinates:

```
this.getUserLocation();
```

The result of searching around me can be some events or nothing:

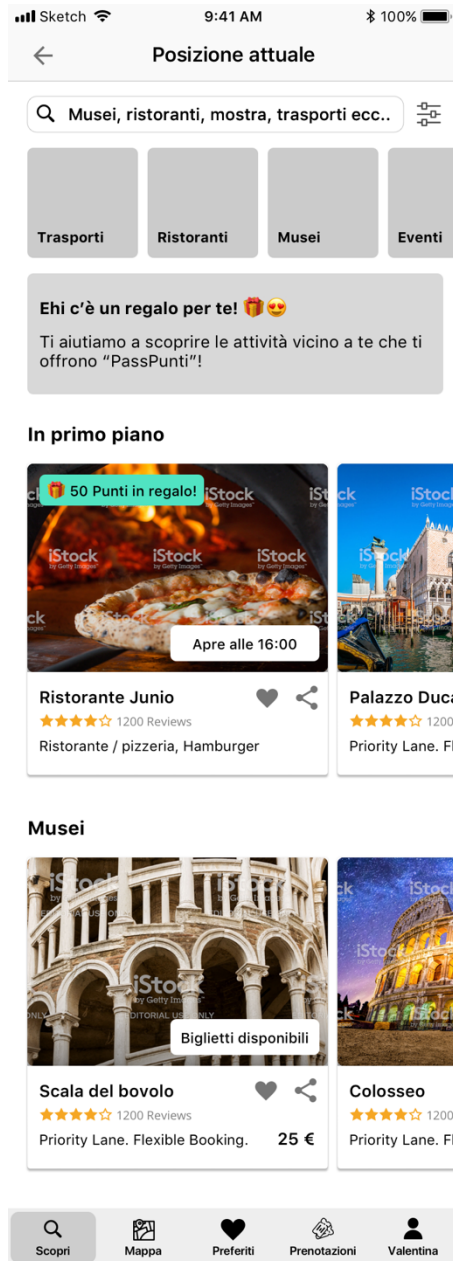


Figure 16: Search page



Spiacente non ci sono ancora attività raggiunte dal servizio MyPass nelle vicinanze!

[Scopri i luoghi dove è attivo il servizio](#)

Figure 17not found

4.3.3. Operations

In this part, there are three sections, that only the privilege user can access to each service, which is for controlling the bought ticket that the controller can check to see if the ticket is valid or not (controlla prenotazione), the other one is for ticket obliteration in the entrance of each event.



Figure 18:operations

In all these sections for reading the tickets, it is needed uses different API, for access to the camera to read the barcode, also there is a library which is used for reading QRCode.

react-native-QRcode-scanner

react-native-camera

the package is installed and imported in the code:

```
import {RNCamera} from "react-native-camera";
```

```
import QRCodeScanner from 'react-native-QRcode-scanner';
```

```
<QRCodeScanner
  cameraStyle={styles.cameraContainer}
  topViewStyle={styles.zeroContainer}
  bottomViewStyle={styles.zeroContainer}
  onRead={this.props.onRead}
  flashMode={RNCamera.Constants.FlashMode.torch}
  flashMode={this.props.torch ?
    RNCamera.Constants.FlashMode.torch :
    RNCamera.Constants.FlashMode.off}
/>
```

Figure 19:QRCode scanner1

```
<QrCodeReader
  active={!this.state.isMenuOpen && !this.state.manual && this.state.listOfPlaces.length > 0}
  torch={this.state.torch}
  onRead={(code) => this.onRead(code.data)}
  readerState={this.state.readerState}
  showReader={this.state.showReader}
  failMessage={this.state.failMessage}
  showError={this.state.showError}
/>
```

Figure 20: QRCode scanner2

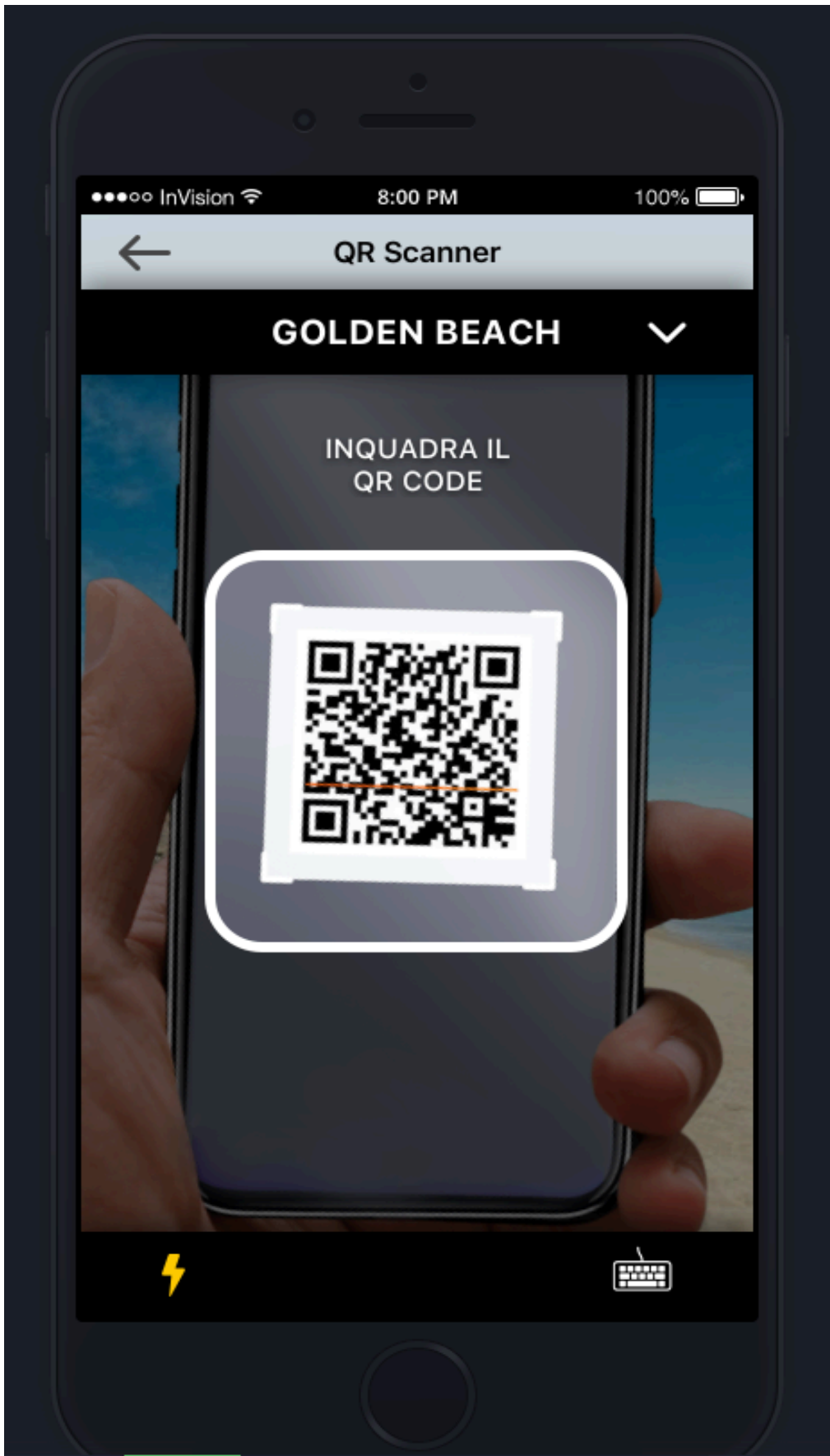


Figure 21: ready scanner

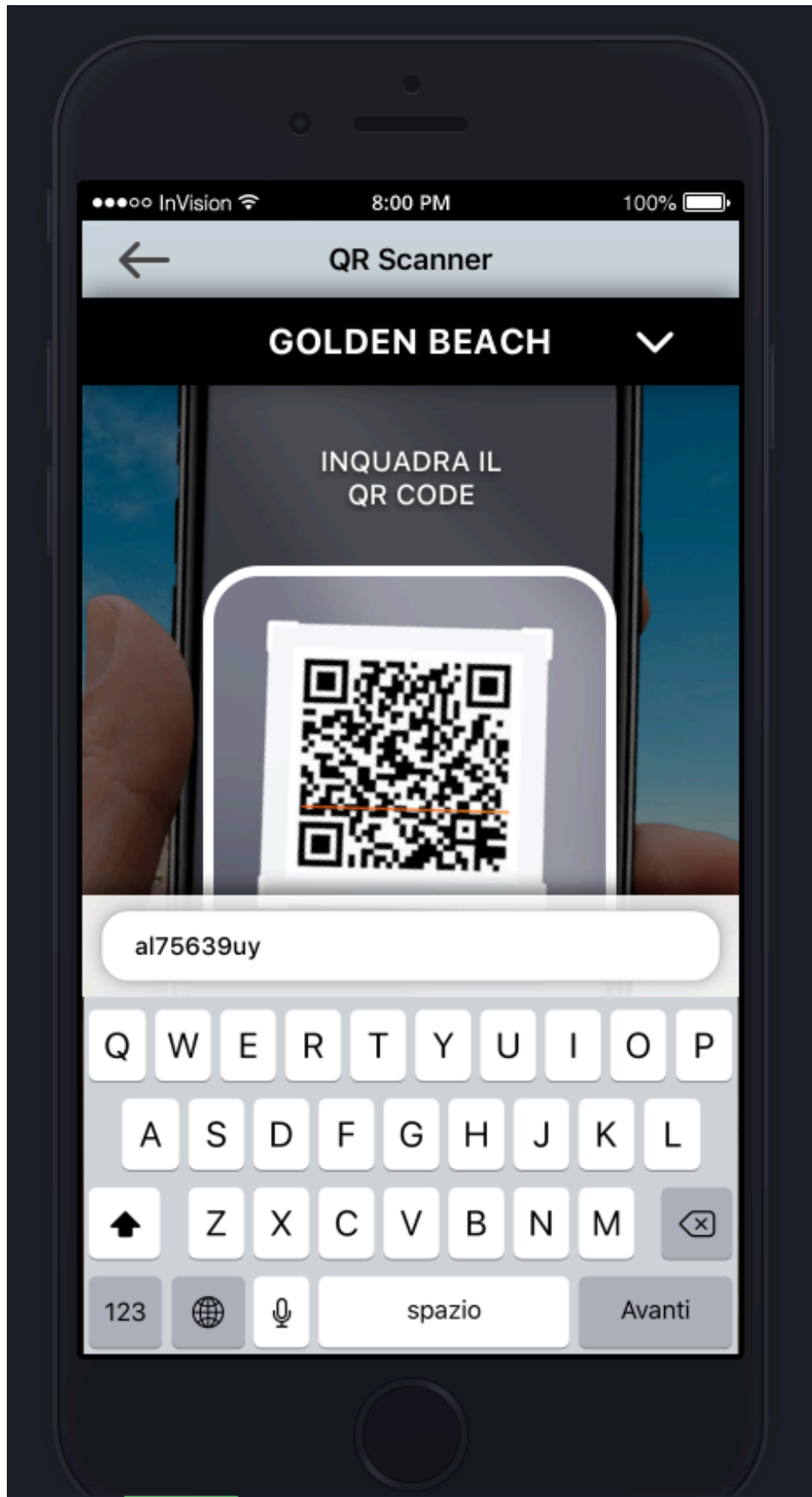


Figure 22: manual insertion

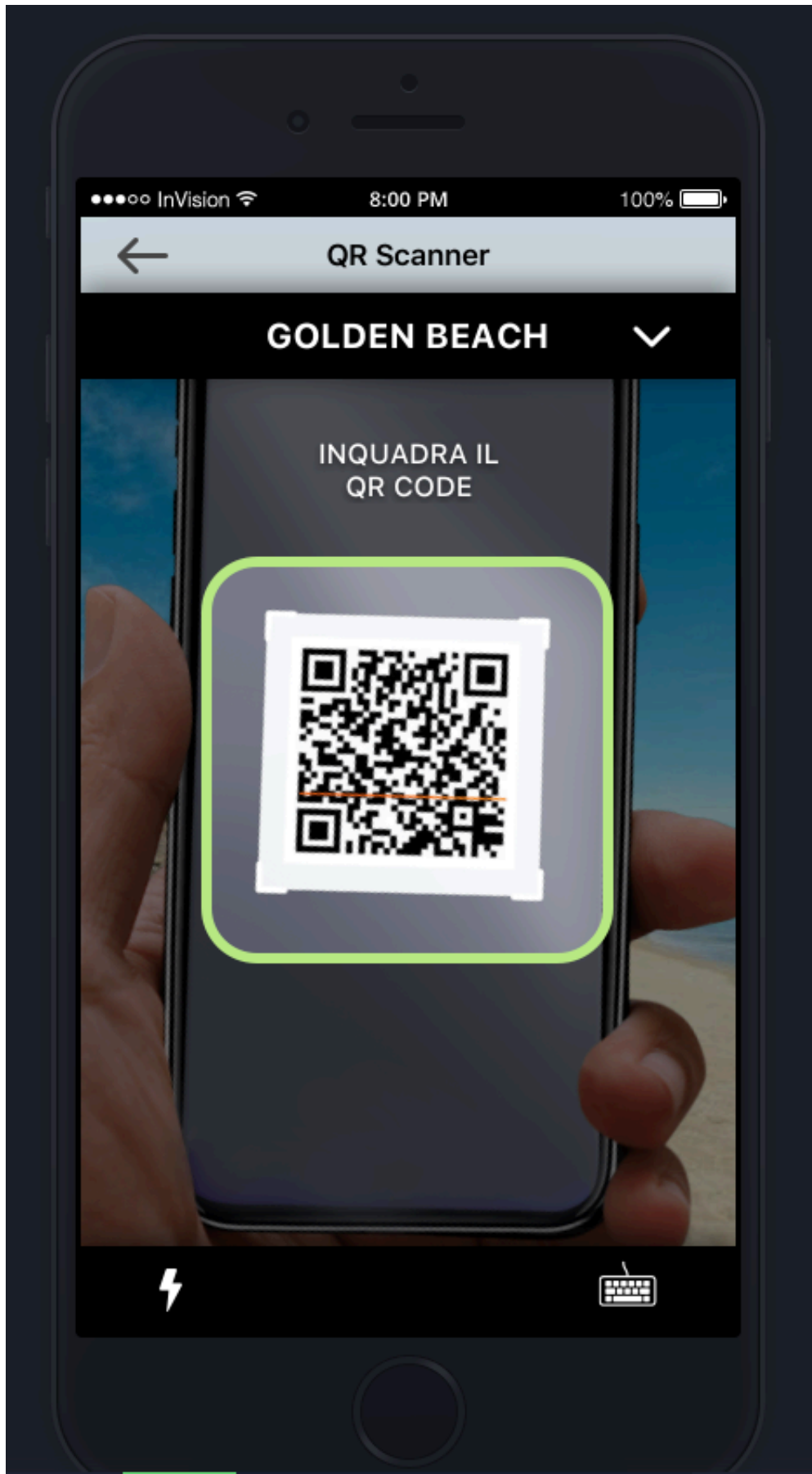


Figure 23: successful

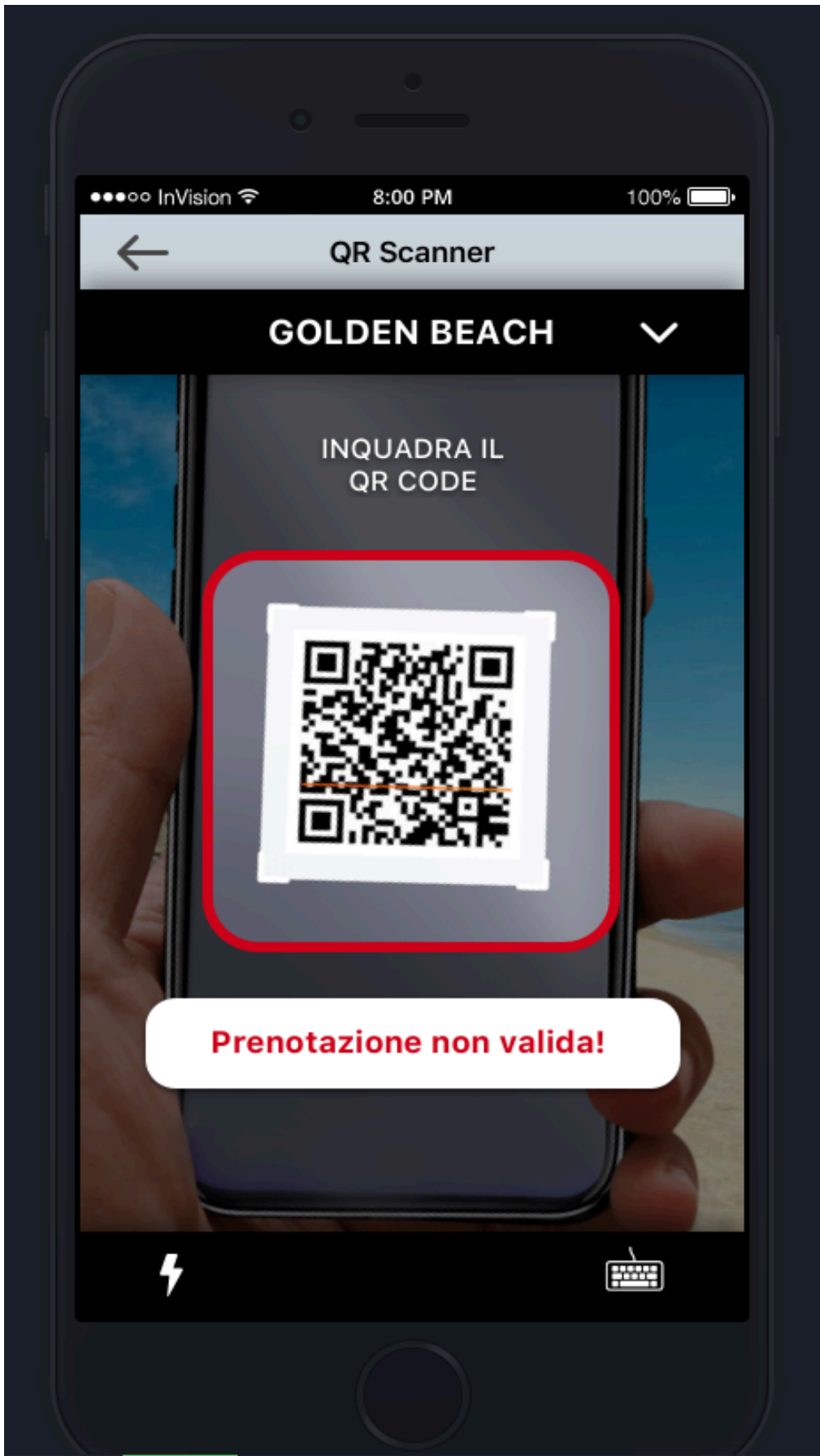


Figure 24: not valid



Figure 25: details

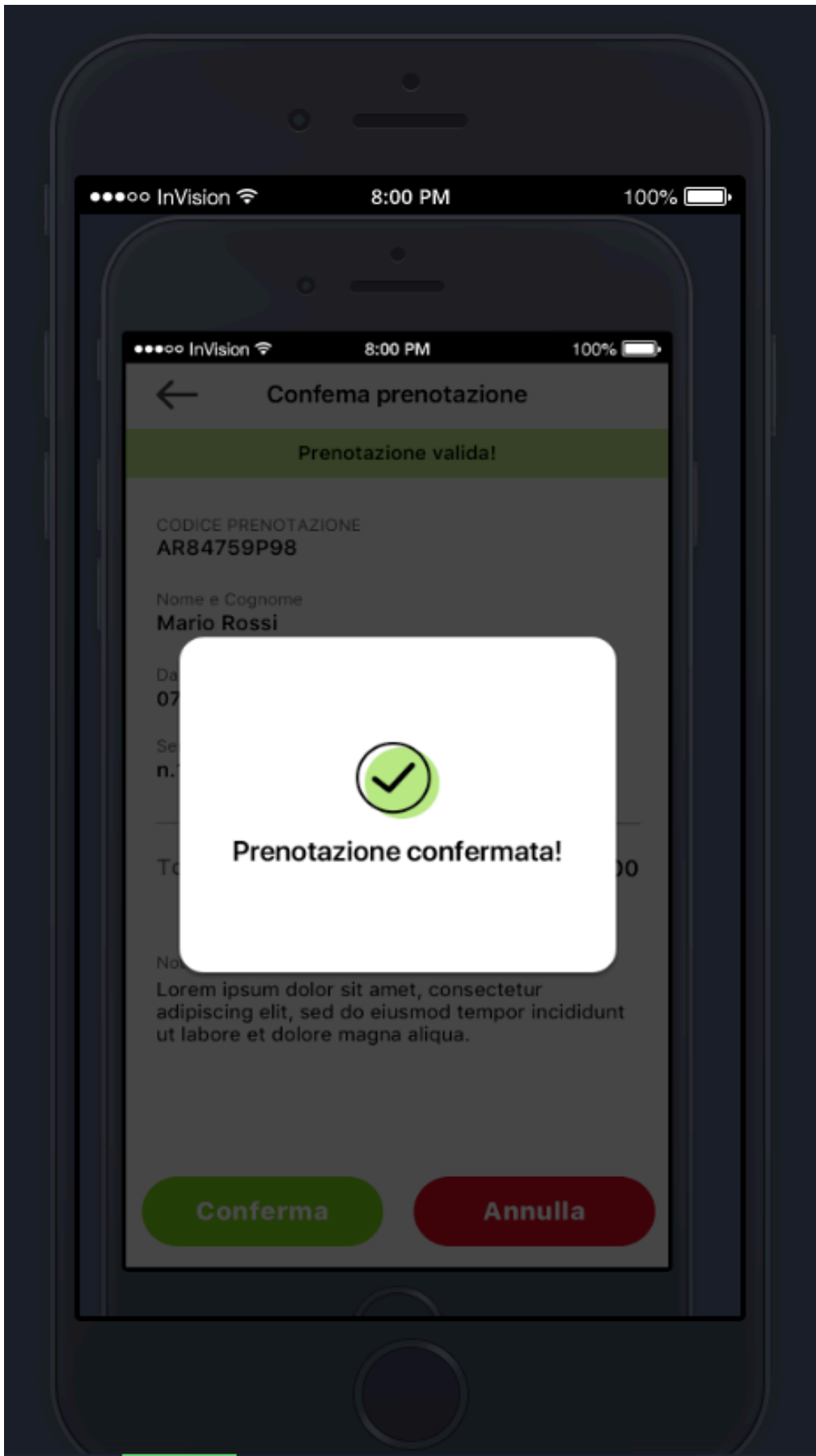


Figure 26: confirmation

4.3.4. Search page

As soon as the user searches according to his requirements based on destination or event, he will be redirected to the search page to see all the results, if the user searches according to the destination, for example, the Venezia is chosen as a destination so all the events based on Venezia will be indicated, also all the details of each event will be shown such as the number of availability, the working hour and all the details and information, so the user can choose what he wants and then continue reserving and shopping tickets by completing the registration in-app.

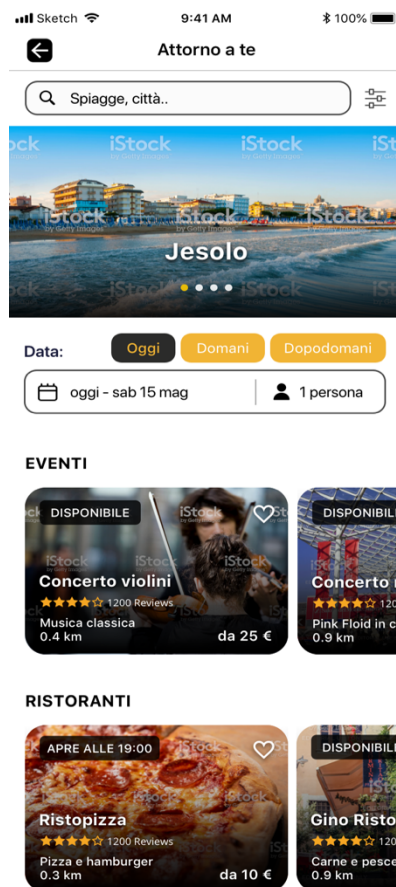


Figure 27: search page 1



Spiagge

🔍 Spiagge, città..



Data:

- Oggi
- Domani
- Dopodomani

📅 oggi - sab 15 mag | 👤 1 persona

CHIOGGIA

DISPONIBILE

Beach 45
★★★★☆ 1200 Reviews
Ombrelloni e ristorazione
0.4 km da 25 €

DISPONIBILE

Chioggia B
★★★★☆ 1200
Ombrelloni e ris
0.9 km

JESOLO

DISPONIBILE

Bagno 30
★★★★☆ 1200 Reviews
Ombrelloni e ristorazione
0.4 km da 25 €

DISPONIBILE

Jesolo Beach
★★★★☆ 1200
Ombrelloni e ris
0.9 km

Figure 28: search page 2

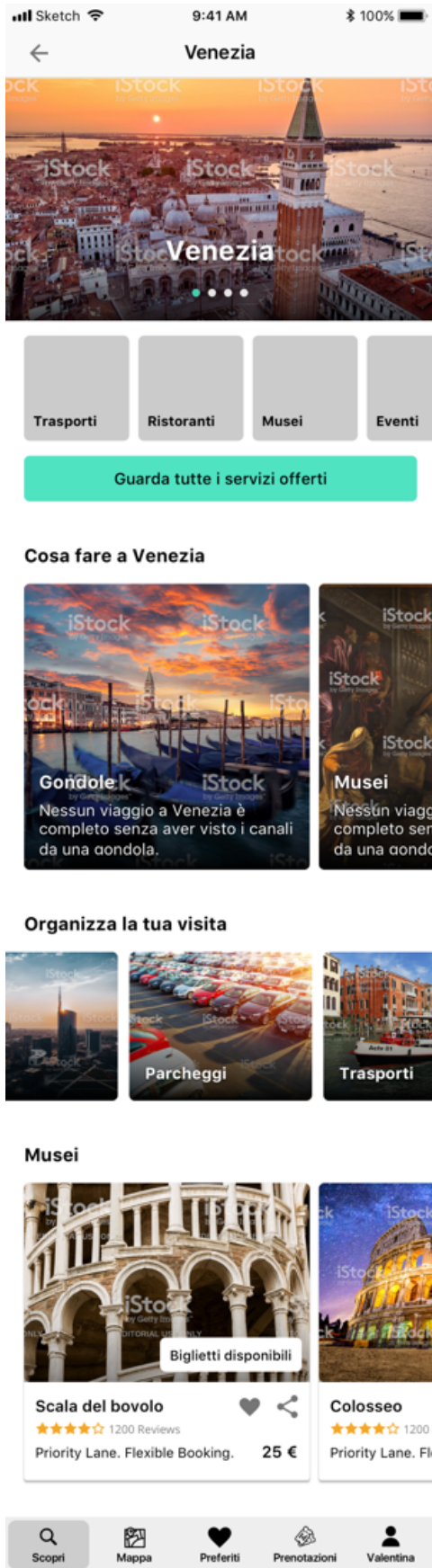


Figure 29:search page 3

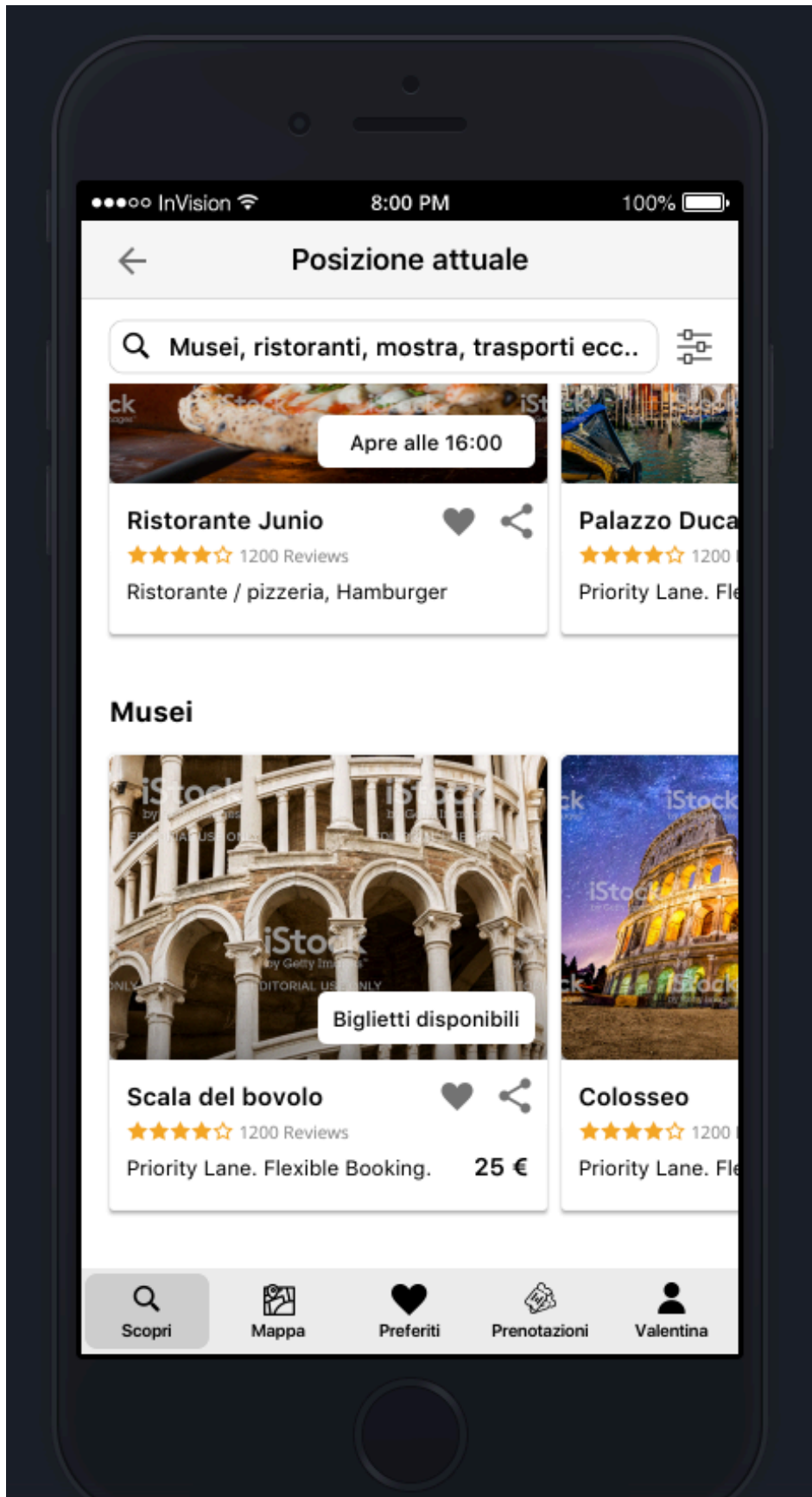


Figure 30: search page 4

The reservation flow is indicated as below:

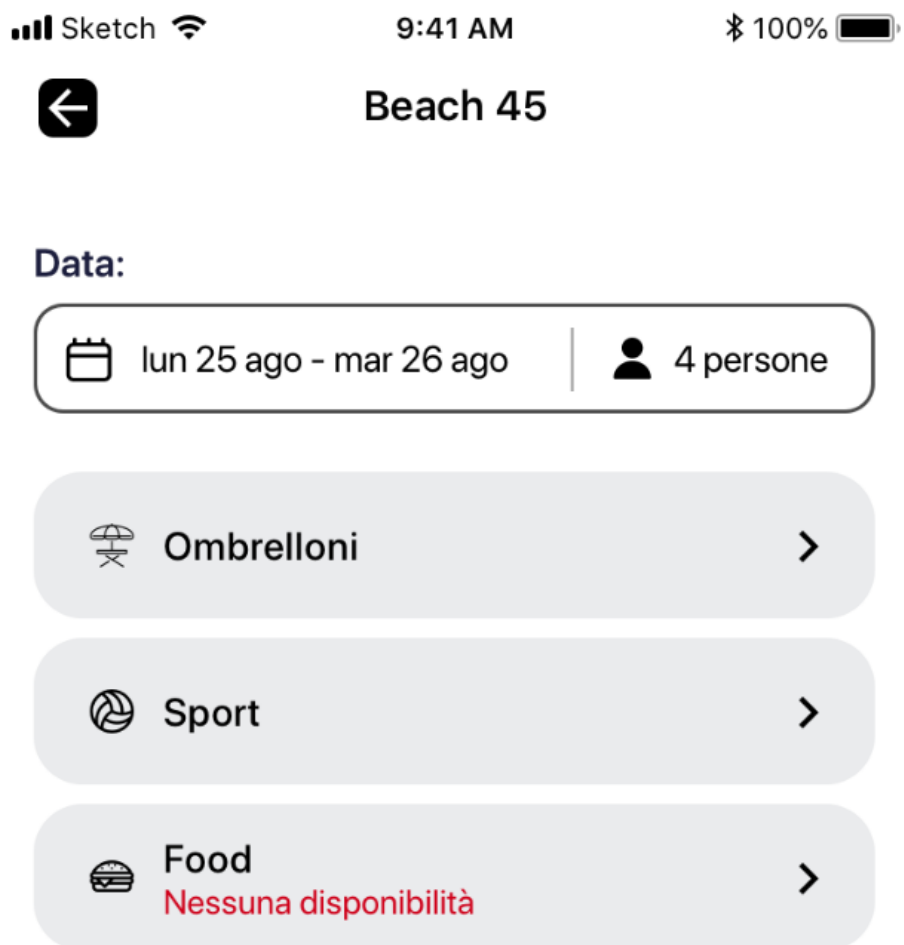


Figure 31: reservation



Ombrelloni

Data:



lun 25 ago - mar 26 ago



4 persone

Scegli

Seleziona il prodotto

Ombrelloni FILA 1 - (3 disponibili)



Ombrelloni FILA 2 - (4 disponibili)



Ombrelloni FILA 3 - (9 disponibili)



Ombrelloni FILA 4 - (13 disponibili)



Vai al riepilogo

Figure 32: availability



Ombrelloni

Data:

lun 25 ago - mar 26 ago | 4 persone

Scegli

Seleziona il prodotto

Ombrelloni FILA 1 - (1 disponibile)

2 ombrelloni x 2giorni - € 100

Scegli

Più giorni - da € 25

Descrizione del servizio: Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Data

Seleziona i giorni

lun 25 ago - mar 26 ago

Quanti?

Seleziona la quantità

2

Capacità massima di X persone per ciascun prodotto

Ombrelloni FILA 2 - (4 disponibili)

Ombrelloni FILA 3 - (8 disponibili)

1 ombrellone x 2giorni - € 50

Ombrelloni FILA 4 - (13 disponibili)

Posti garantiti per 10 minuti dalle (ora)!

Totale:

€ 150,00

Vai al riepilogo

Figure 33: reserve details



Conferma acquisto

Riepilogo

Controlla tutti i dati e procedi con il pagamento.

Bagno Beach 45 Chioggia
Lunedì 25 agosto - Martedì 26 agosto

- 2 Ombr. 1 FILA x 2 giorni € 100,00
- 1 Ombr. 3 FILA x 2giorni € 50,00

Specifica i dati di riferimento corretti per eventuali contatti da parte del fornitore.

Numero di telefono

+39

Richieste/note aggiuntive

Comunica qui ulteriori richieste non specificate in precedenza



Aggiungi prodotto

Totale: € 150,00

Le informazioni sono trasmesse tramite connessione sicura.



Figure 34: payment



Beach Volley

Data:

Calendar icon | lun 25 ago | 4 persone

Scegli

Seleziona il prodotto

Beach Volley ▼
 Campo 1 - 09:30 / 11:00

Scegli

Fascia oraria - 6€/ora ▼

Descrizione del servizio: Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Data
 Seleziona i giorni

Calendar icon | lun 25 ago

Per quando?
 Seleziona orario

09:30 ▼ | 10:30 ▼

Quanti?
 Seleziona la quantità

- 1 +

Capacità massima di X persone per ciascun prodotto

Pedalò ▼

Paddle ▼

Posti garantiti per 10 minuti dalle (ora)!

Totale: € 25,00

Vai al riepilogo



Conferma acquisto

Riepilogo

Controlla tutti i dati e procedi con il pagamento.

Bagno Beach 45 Chioggia Lunedì 25 agosto - Martedì 26 agosto

- 2 Ombr. 1 FILA x 2 giorni € 100,00
- 1 Ombr. 3 FILA x 2giorni € 50,00

Bagno Beach 45 Chioggia Lunedì 25 agosto

- 1 Campo B.Volley x 2 ore € 25,00

Specifica i dati di riferimento corretti per eventuali contatti da parte del fornitore.

Numero di telefono

Italy flag +39 333 56789354

Richieste/note aggiuntive

Comunica qui ulteriori richieste non specificate in precedenza



Aggiungi prodotto

Totale: € 175,00

Le informazioni sono trasmesse tramite connessione sicura.

Apple Pay

Paga

Figure 35: Confirm payment

4.3.5. Login

it is implemented three login ways to the app, google login, Facebook login, and log in with email.

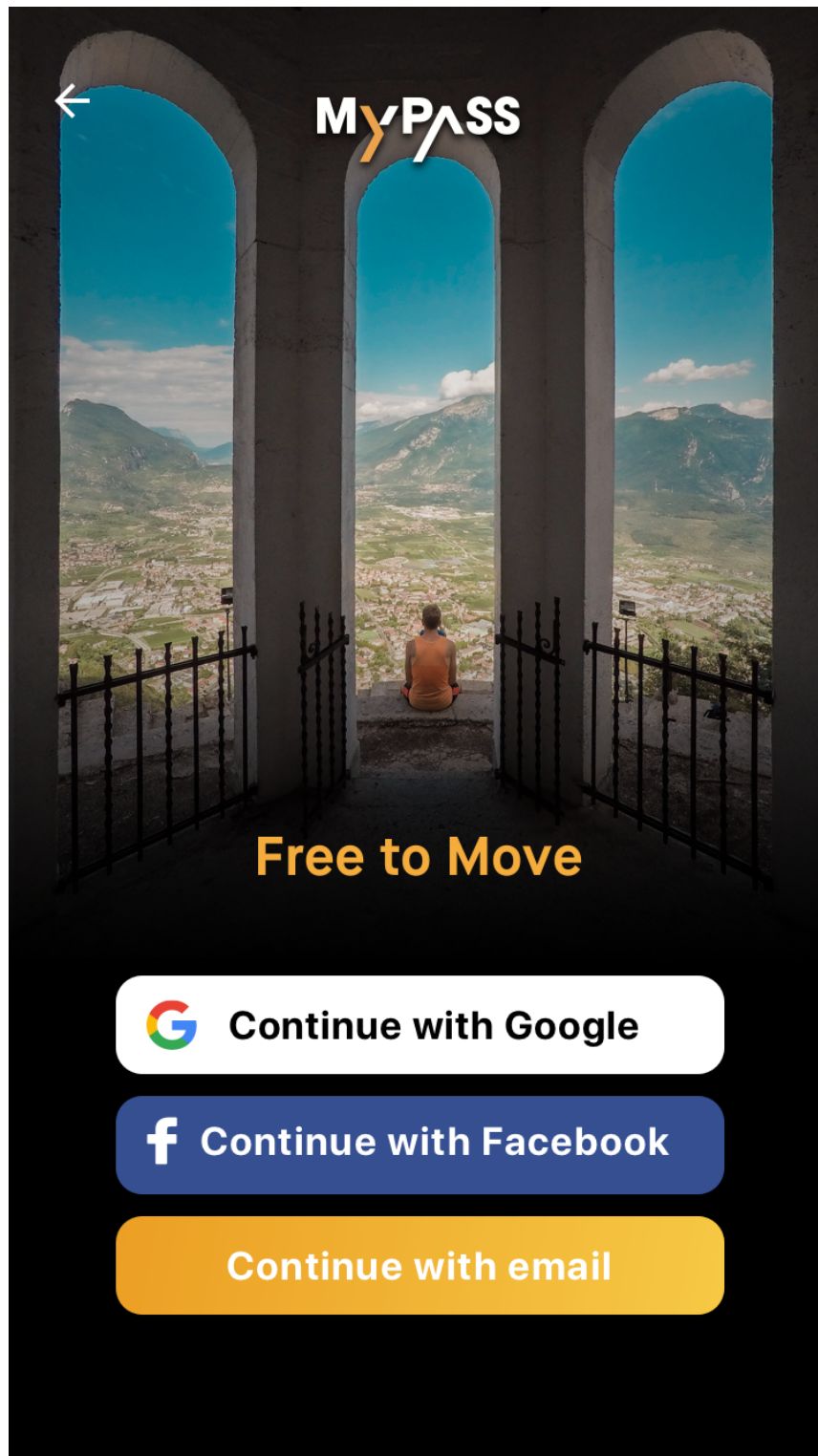


Figure 36: Login screen

Why it is useful to have third parties login in the app?

- 1) You can make your authentication process simple and friendly with Google or other third parties. The registration process doesn't have to waste time, which will greatly increase the registration and retention rates.
- 2) It's safe and secure.
- 3) Users rely more than an unknown website or app on Google or Facebook on the Internet.
- 4) It offers a good experience for users. We as users have little patience for any acts or function, particularly in a relatively unknown app, we first try.

4.3.5.1. Google Login

Google login is an incredible feature which makes the app easier for the user to create an account and sign in.

also, it has some benefits for the developer that firebase makes it super easy for them to add support for google sign in.

react-native and firebase make the implementation of google login simple and without any difficulty.

Here the process of Google login implementation is described as below:

First, we should install the `react-native-google-sign in` the package for implementing the google auth functions in react native then we should import all the required modules and components from this package as shown below:

```
import {
  GoogleSignin,
  GoogleSignInButton,
  status codes,
} from 'react-native-google-sign in';
```

Next, we need to create some states for handling the auth state and user info:

```
const [loggedIn, setloggedIn] = useState(false);
const [userInfo, setUserInfo] = useState([]);
```

Then, the sign-in function should be created in order of handling the authentication:

```
_signIn = async () => {
  try {
    await GoogleSignin.hasPlayServices();
    const {accessToken, idToken} = await GoogleSignin.signIn();
    setloggedIn(true);
  } catch (error) {
    if (error.code === statusCodes.SIGN_IN_CANCELLED) {
      // user cancelled the login flow
      alert('Cancel');
    } else if (error.code === statusCodes.IN_PROGRESS) {
      alert('Signin in progress');
      // operation (f.e. sign in) is in progress already
    } else if (error.code === statusCodes.PLAY_SERVICES_NOT_AVAILABLE) {
      alert('PLAY_SERVICES_NOT_AVAILABLE');
      // play services not available or outdated
    } else {
      // some other error happened
    }
  }
}
```

```
}  
}  
};
```

In this step we should initialize the setup of google login object with taking advantage of the `useEffect` function:

```
useEffect(() => {  
  GoogleSignin.configure({  
    scopes: ['email'], // what API you want to access on behalf of the user, default is email and profile  
    webClientId:  
      '418977770929-g9ou7r9evalu78a3anassxxxxxxx.apps.googleusercontent.com', // client ID of type  
      WEB for your server (needed to verify user ID and offline access)  
    offlineAccess: true, // if you want to access Google API on behalf of the user FROM YOUR SERVER  
  });  
}, []);
```

Last but not least, we need a function for controlling the logout action, so the `signOut` method will be used for this purpose:

```
signOut = async () => {  
  try {  
    await GoogleSignin.revokeAccess();  
    await GoogleSignin.signOut();  
    setloggedIn(false);  
    setUserInfo([]);  
  } catch (error) {  
    console.error(error);  
  }  
};14
```

4.3.5.2. Facebook Login

A) Check login status of the user

First, we should check the login status of the user to see if he is already logged-in with Facebook or no, for this purpose we should have a call to [FB.getLoginStatus](#), then Facebook calls back with the results:

Sample Call :

```
FB.getLoginStatus(function(response) {  
    statusChangeCallback(response);  
});
```

Sample JSON Response :

```
{  
    status: 'connected',  
    authResponse: {  
        accessToken: '{access-token}',  
        expiresIn: '{unix-timestamp}',  
        reauthorize_required_in: '{seconds-until-token-expires}',  
        signedRequest: '{signed-parameter}',  
        userID: '{user-id}'  
    }  
}
```

```
}
```

The status describes the login status of the user, which can be:

Status Type	Description
<code>connected</code>	The person is logged into Facebook, and has logged into your webpage.
<code>not_authorized</code>	The person is logged into Facebook, but has not logged into your webpage.
<code>unknown</code>	The person is not logged into Facebook, so you don't know if they have logged into your webpage. Or <code>FB.logout()</code> was called before, and therefore, it cannot connect to Facebook.

Figure 37: login status

IF the status is `connected`, the following `authresponse` parameters are included in the response:

<code>authResponse</code> Parameters	Value
<code>accessToken</code>	An access token for the person using the webpage.
<code>expiresIn</code>	A UNIX time stamp when the token expires. Once the token expires, the person will need to login again.
<code>reauthorize_required_in</code>	The amount of time before the login expires, in seconds, and the person will need to login again.
<code>signedRequest</code>	A signed parameter that contains information about the person using your webpage.
<code>userID</code>	The ID of the person using your webpage.

Figure 38: response

B) Log a user in

If the user opens the app and he is not logged in or not logged in to Facebook, we should use the login dialog to force them to login into both. If they are not logged into Fb, they will first be asked to log in, then asked to log in to the app.

C) Log a user out

when the user wants to log out we should attach JavaScript SDK function `FB.logout()` to a button or a link.¹⁵

```
FB.logout(function(response) {  
    // Person is now logged out  
});
```


4.4. Backend APIs

In this project, we use 2 different kinds of server, one of them is a physical server which is in Italy and the other one is a virtual server which is implemented on AWS.

The responsibilities of our backend team are writing APIs, writing code to interact with the database, creating libraries, working on data architecture.

Backend development enables the app to process the actions that users on the frontend take and deliver the correct information to retrieve.

The backend team collect all our APIs in a swagger that every member of the team can access to it and know what is the functionality of each API.

4.4.1. Branch

We have two main branches:

TEST branch: we use it for our internal development, while we are developing our app for testing all the APIs functionality.

PRODUCTION branch: this branch is used in case of releasing the app and we have RestCall to retrieve the information and show it in the app.

4.4.2. PHP

PHP is the second server-side programming language we know.

for creating the server backend, we always use PHP, we use PHP to provide data, and some times in the frontend reads some user data from a database and returns it in JSON format, also, we can collect form data, generate our home page application dynamically.

4.4.3. AWS

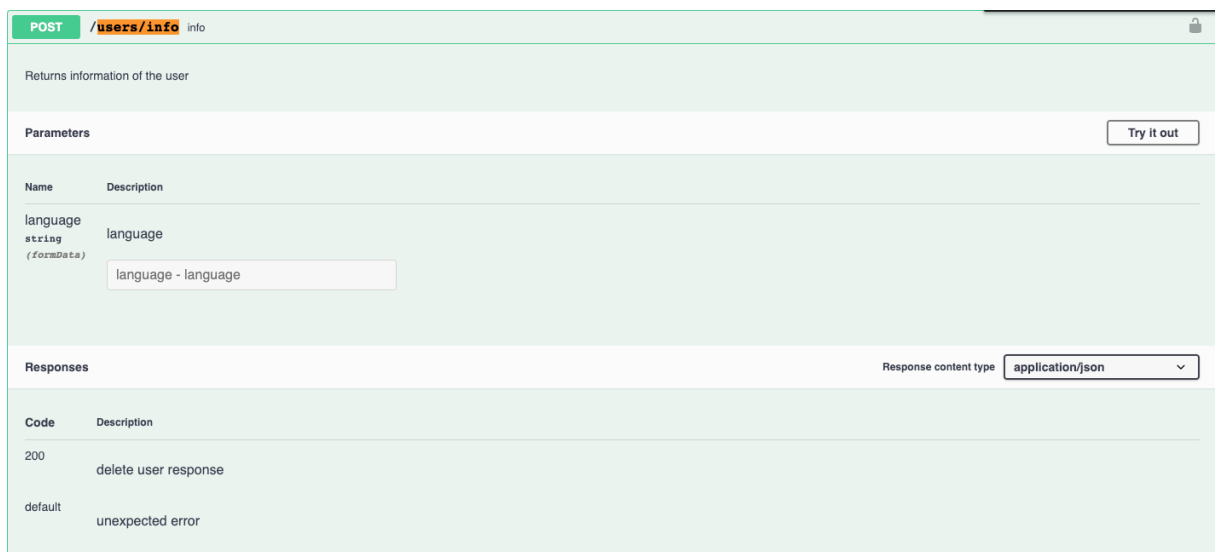
The company's goal is to accelerate the response of the server, so the backend team is going to use AWS Lambda for serverless computing, so they only write codes that serve user, so they do not care about infrastructure management task like capacity provisioning because it will handle with AWS.

4.4.4. APIs and RestCalls

All our APIs which are provided from the backend team will be organized in swagger so we can have access to it and use them according to our need, I will explain some principal RestCalls.

userInfo

in the homepage of the application, once the user enters the app, we need a POST call to the server to retrieve all the user information, like name, surname, profile photos and more other information.



POST /users/info info

Returns information of the user

Parameters Try it out

Name	Description
language string (FormData)	language
	<input type="text" value="language - language"/>

Responses Response content type: application/json

Code	Description
200	delete user response
default	unexpected error

```
export function userInfoFetch(that, access_token) {
  var params = {};
  params.portal = Config.portal;
  params.platform = Config.platform;
  params.language = Config.language;
  params.version = Config.version;
  let url = Config.hostUrlV1 + 'users/info'

  console.log('calling rest @', url, access_token);
  that.setState({ isLoading: true });
  return fetch(
    url,
    {
      method: 'POST',
      headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
        'Authorization': 'Bearer ' + access_token,
      },
      body: JSON.stringify(params),
    }
  )
}
```

```

.then((response) => checkStatus(that, response))
.then((response) => {
  console.log('response info:', response);
  that.setState({ isLoading: false });
  // console.log("[response] userInfoFetch", response);
  if (response.name) {
    var name = response.name;
    name = name.charAt(0).toUpperCase() + name.slice(1);
    var surname = response.surname;
    surname = surname.charAt(0).toUpperCase() + surname.slice(1);

    Storage.saveItems({
      logged: true,
      user_id: response.id_ut,
      name: name,
      surname: surname,
      phone: response.tel_cell,
      email: response.email,
      birthday: response.data_nasc,
      photo: response.photo,
    });
  }

  return response;
})
.catch(e => {
  that.setState({ isLoading: false });
  //console.log('Network exception: ', e);

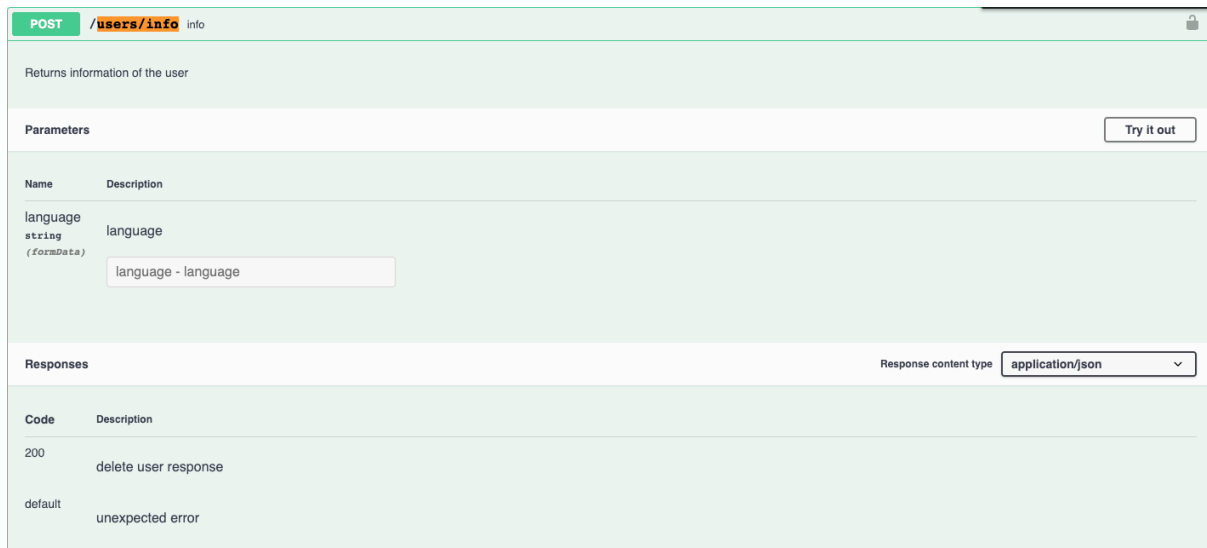
  if (e.message.indexOf("Network request failed") >= 0) {
    DataManager.getInstance().setConnectionState(false);
    DataManager.getInstance().getUser();
  }
  return false;
});
}

```

The Bearer Token is used for authorization, so with calling the server all the user information is returned and we will save it in the local storage in order to use them in offline mode, or in other parts of apps.

jsonEvents

we have a very long JSON which returns data and will display in the given portal, it means that some parts of the app will be filled dynamically with all the information in JSON.



The screenshot shows a REST client interface for a POST request to `/users/info`. The request description is "Returns information of the user". The parameters section shows a single parameter named "language" of type "string" (FormData), with a text input field containing "language - language". The responses section shows a table with two entries: a 200 status code for "delete user response" and a default status code for "unexpected error". The response content type is set to "application/json".

Name	Description
language string (FormData)	language <input type="text" value="language - language"/>

Code	Description
200	delete user response
default	unexpected error

```
export function jsonEvents(that, params) {
  params.portal = Config.portal;
  params.platform = Config.platform;
  params.language = Config.language;
  params.version = Config.version;
  let url = Config.hostUrlV1 + 'event/jsonEvents' + objectToURIParams(params);

  console.log('calling rest @', url, params);
  that.setState({ isLoading: true });
  return fetch(
    url,
    {
      method: 'GET',
      headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
        'Authorization': 'Basic ' + Config.hostBasicAuth,
      },
      // body: JSON.stringify(params),
    }
  )
  .then((response) => checkStatus(that, response))
  .then((response) => {
    //console.log('✓ ' + arguments.callee.toString().match(/function
    ([^\\(]+)/)[1], 'response:', response);
    that.setState({ isLoading: false });
    return response;
  })
  .catch(e => {
```

```
that.setState({ isLoading: false });
//console.log('Network exception: ', e);

if (e.message.indexOf("Network request failed") >= 0) {
  DataManager.getInstance().setConnectionState(false);
  return null;
}
});
}
```

getCreditCardInfo

returns all the pieces of information related to the user credit card.

let url = Config.hostUrlV1 + 'payment/getCreditCardInfo' + objectToURIParams(params);

checkAppVersionExt

Function check if the version is fine according to app_locks table.

let url = Config.hostUrlV1 + 'users/checkAppVersionExt' + objectToURIParams(params);

5. Chapter 5

5.1. Conclusion

Nowadays using the new technologies and the different apps is a trend, and people all over the world will do at least one purchase with their mobile phone, and as it is obvious using desktop purchase will be replaced with mobile commerce, because of accessibility and acceleration in operations.

People are more willing toward mobile using, apart from reviewing, giving feedback, and price comparison they also have started to analyze the overall user experience of any mobile app.

So, all the people trying to have experiences with different kind of apps, especially those kinds of apps which save the people time, and enjoyable to work with them.

since covid-19 pandemic is all over the world, people prefer to use the reserve app in every place, in order to skip the line and have less contact with each other.

So MyPass App is the best practice for this purpose, which helps people to check all the available places, the working time and they can decide to choose destination so the ticket can be reserved and bought.

MyPass app is one of the robust apps in Italy which covers all the user requirements.

5.2. Limitation and Future works to improve application

5.2.1 Adding live chat assistance:

Currently, in Mypass App we have a phone number and also email address so customers can call us in case of any problem. we have customer service in our office who cares about client issues, but the company has a goal to add a new part in the app which is **live chat assistance** so the users can communicate with customer service in real-time and without waiting in queue for calling.

5.2.2 Payment methods:

We have apple pay in our application to make payment easy, but still, need to improve the payment method so in the future we will also add PayPal and satisfy payment, which is so popular nowadays.

5.2.3 Biometric Authentication:

Mobile application biometric authentication is a multi-factor authentication (MFA) approach to verify the identity of a person who uses possession of a mobile device as a first factor and uses that application to verify a unique biometric identifier as a second factor.

Passwords are not as safe as they were once. As cybersecurity is becoming an increasingly hot problem with infractions now happening regularly, new ways to secure data can be used to log in to mobile apps. One of these approaches is the use of mobile biometric authentication.

Biometric authentication is fast, safe and easy to use. Users love fingerprint or facial authentication because these authentication mechanisms allow them to access their devices safely and with minimal effort.¹⁶

So, we want to add fingerprint and face recognition in the future to make our app more user friendly.

5.2.4 Cloud servers:

Most of the company's servers are physical and they are in Italy, we do not have any problem now because our app only is used inside Italy, but as I mentioned the goal of the company is to expand the app all over the world and can be used for different events around the world, but as our servers are in Italy if a user wants to use our app in the united states so he faced with a problem because the app will be very slow in response to user due to server delay, so to improve this problem and in order to accelerate the response to the user we will integrate our server in the cloud and uses cloud services like Aws.

5.2.5 Makes app simpler:

Since people using mobile devices in busy environments, and they push their patience and focus down. many people are still so annoyed by the difficulty of mobile apps and they only leave and uninstall the app without ever giving a second thought, so to have the best user experience, we should simplify our app from product discovery to payment. I mention some tips for improvement:

- we should use easy navigation.
- engage app experience with more photos and fewer texts.

Bibliography

- ¹ <https://www.techopedia.com/definition/22193/software-development-life-cycle-sdlc>
- ² <https://www.sciencedirect.com/topics/computer-science/system-development-life-cycle>
- ³ <https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/>
- ⁴ https://www.tutorialspoint.com/sdlc/sdlc_overview.htm
- ⁵ <https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/>
- ⁶ <https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/>
- ⁷ <https://performancelabus.com/software-testing-importance-sdlc/>
- ⁸ <https://textexpander.com/blog/7-stages-of-the-system-development-life-cycle/>
- ⁹ <https://www.thedigitalmentor.com/what-is-maintenance-in-sdlc/>
- ¹⁰ <https://javascript.info/intro>
- ¹¹ <https://help.apple.com/xcode/mac/current/#/devc8c2a6be1>
- ¹² <https://reactnative.dev/>
- ¹³ https://en.wikipedia.org/wiki/Amazon_Web_Services
- ¹⁴ <https://www.freecodecamp.org/news/google-login-with-react-native-and-firebase/>
- ¹⁵ <https://developers.facebook.com/docs/facebook-login/web>
- ¹⁶ <https://medium.com/@berina.omerasevic97>