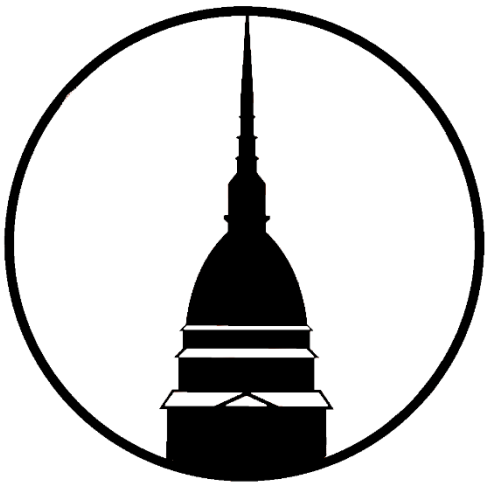
POLITECNICO DI TORINO

PROJECT AND LABORATORY ON

EMBEDDED COMMUNICATION SYSTEMS

-PROJECT REPORT-

EventsByPolito



Students:
Braccio Jacopo (s273999)
Naggi Giulio (s269112)
Polidori Brendan David (s274990)

Professor:
Guido Albertengo (guido.albertengo@polito.it)

Academic year 2019/2020

Contents

1. Introduction

The proposed project aims at the creation of an event booking platform. The platform will have a catalogue with different kinds of events (musical, cultural, movies, etc.) inserted by some promoters. Clients can access the catalogue through different devices (mobile phone, personal computer, dedicated terminal) and book/buy tickets for the events in which they are interested in. Once the ticket is bought, it will be virtually available to the user who can then scan it through a custom-made gate, located in the event's site, in order to access the event.
The following project can be useful in the current global health situation: the possibility of buying tickets remotely, without the need of physically being in the event's place, and the gate scanning system could help avoiding crowds and therefore reducing the spread of COVID-19.
The name of the project "EventsByPolito" is chosen.

2. Project description

2.1 Actors' classification

Actors are external entities that interact with the systems. Four kind of actors can be identified for the proposed project, each one with different roles and capabilities:

- ***Simple user***: is a user that is not registered to the platform. It is able only able to look at the catalogue of the events and register to the platform;
- ***Registered user***: is a user registered to the platform. It can place reservations to events and pay for them, contact the promoter and the platform manager;
- ***Promoter***: is the entity that creates events. It can be a normal person, an association or a club. It is responsible of everything regarding the events that it has created, such as giving the correct information and payments;
- ***Staff member***: is a person entitled by the promoter whose role is to help during the event. It can scan customers' tickets and let them access the event or not;
- ***System manager***: is the entity managing the entire platform (website, web-server, apps, etc).

2.2 Requirements elicitation

2.2.1 Functional requirements

Functional requirements define the basic system behavior, making clear software functionality that must be implemented in order to satisfy user's needs. The functional requirements met in developing the platform are described in TABLE A.

Table A *- Functional requirements: SU (simple user), RU(registered user), P (promoter), S (staff).*

REQ. ID	DESCRIPTION
Provider_01	Registration and access to the platform
Provider_02	Creation of events: each event should contain a description, information about tickets' pricing, time access slot and where the event takes place.
Provider_03	Define different kind of tickets with different price
Provider_04	Check the reservations received for each event
Provider_05	Check live status of each event (empty and occupied seats)
Provider_06	Elect member of its staff
Prover_07	Modify the event
Staff_01	Scan tickets through QR code via mobile app
SimpleUser_01	Access events' catalogue from web, app and totem
SimpleUser_02	Check information about a single event
SimpleUser_03	Register to the site
RegisterdUser_01	Login to the platform
RegisterdUser _02	Make reservation from web platform, mobile app and totem
RegisterdUser _03	Online payments
RegisterdUser _04	History of purchases
RegisterdUser _05	Contact event's promoter
RegisterdUser _06	Receive a virtual ticket via email
RegisterdUser_07	Contact platform administrator
RegisteredUser_08	Share interesting events
RegisterdUser_09	Gate check
ALL_01	Multi Language support

2.2.2 Nonfunctional requirements

The platform is web-based and it can be accessed from personal computers, mobile phones (Android and iOS) and through a custom-made terminal that will be positioned in certain points of the city. In order to guarantee the communication between devices, some web services (in particular RESTful APIs) have been developed. APIs calls and data are stored inside a Linux web server whose functions are developed using a Python micro-framework for web development called Flask including a PostgreSQL database.

2.3 Use case diagram

Figure 1 represents the use case diagram of the system, that is a visual representation of what actions the main actors can perform. '*Registered user*' is a generalization of '*Simple user*' therefore it can perform all of its actions.

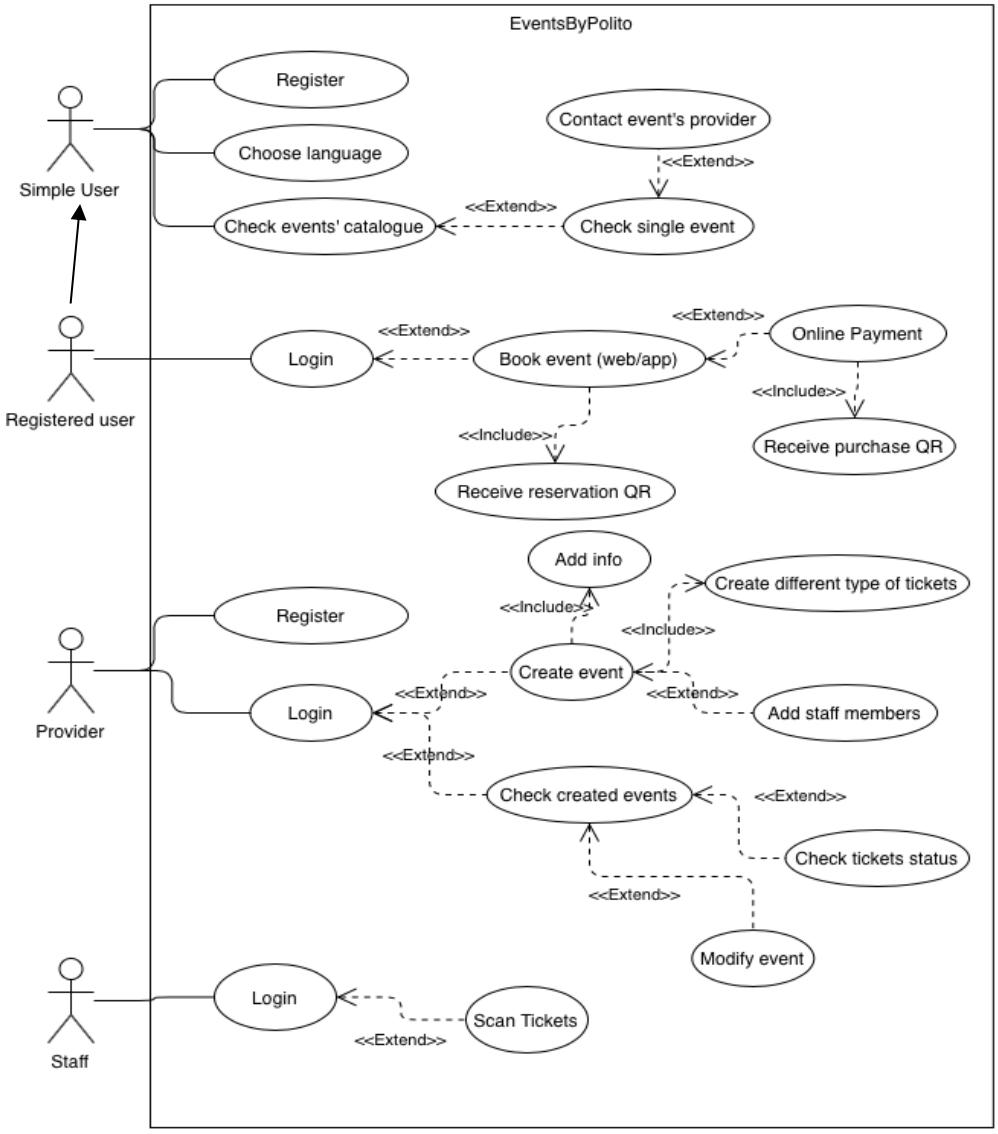


Figure 1 - Use case diagram.

3. Description of the system

3.1 Backend server

The backend server is installed on a virtual machine and runs “Ubuntu 20.04.1 LTS” Linux distro. It is used to manage the web platform, Flask-based, and the database where all the data are stored. The former is handled by the Nginx reverse proxy that receives the requests and redirects them to the Gunicorn web server which is a WSGI python-based software that is able to administrate the Flask web application. The database is provided by the Postgresql software and it is directly connected to the Flask application.

For what concerns the API requests, the management is provided by the Flask application that elaborates the data and sends a JSON response to the requesting application.

The web platform is reachable both with the HTTP and the HTTPS protocol, since a Let’s encrypt SSL certificate has been installed to allow payments.

3.2 Front end web platform

The web platform is developed using the micro framework Flask. It allows for the implementation of web pages that are created using the jinja module which interacts with the python code. The modules are separated into blueprints which makes the web platform modular, allowing for easy implementation of new features. Each blueprint takes care of one feature of the web platform, allowing for communication between the necessary information and the client web view. SQLAlchemy is used as an interface to the database, and allows for easy access to information and operations. SQLAlchemy also allows for the table characteristics to be defined in python without the need for SQL. Payments are done through Stripe™, which allows to accept credit cards and also gives the opportunity for further expansion to accept Apple Pay and Google Pay.

All the information is displayed to the client side by using templates, which are html pages with integrated jinja variables, that allow for dynamic values on a static html page. Also Javascript is used in order to render the user experience more simple and straightforward. The Javascript files are stored under the static directory which also contains the folders for all the images, pdfs and information. Emails are also implemented through ‘secure smtp ssl’, that allows a user to receive an email containing the information needed once an order is completed. It is also possible for the user to download specific bookings in pdf form.

Client loggin into the web platform can perform the following main actions:

- Access events' catalogue (Figure 2);
- Look at a detailed list of info for each event (Figure 3);
- Look at the different types of tickets available for each event;
- Place a reservation;
- Pay a reservation using StripeTM (Figure 4);
- Get a virtual ticket and receive it via emai (Figure 5);
- Look at the past reservations;
- Contact the event's provider.

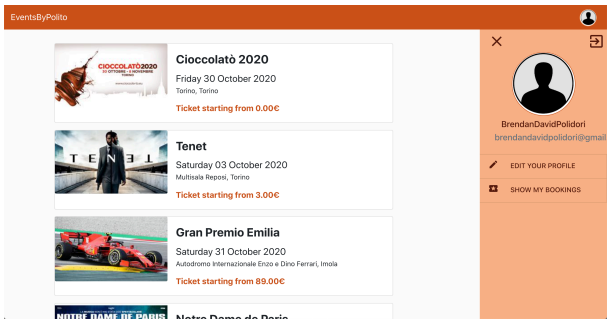


Figure 2- Events' catalogue

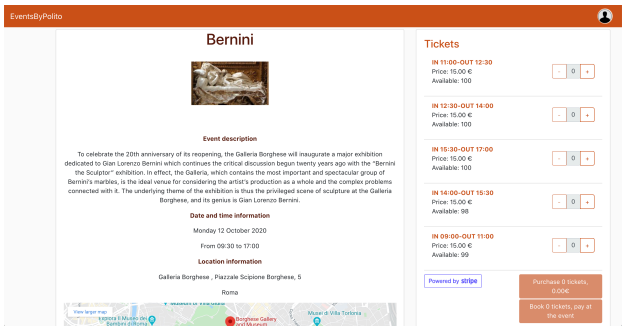


Figure 3 – Single event's page

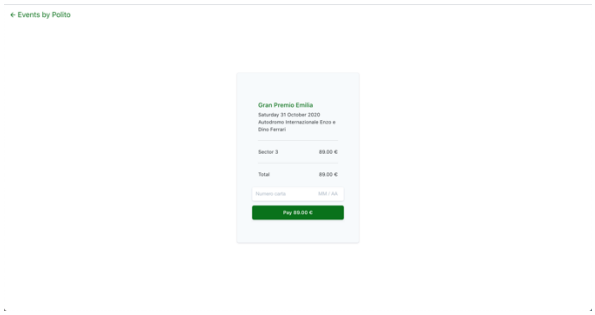


Figure 4- Stripe payment view

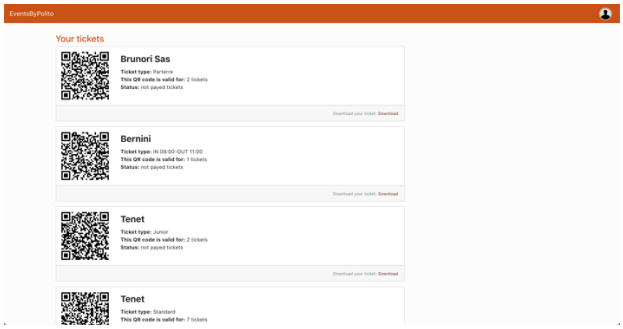


Figure 5 – Booking and virtual ticket section

Provider loggin into the platform can perform the following main actions:

- Look the list of events it has created (Figure 6);
- Updated the events already created;
- Add new events by adding all the necessary info (Figure 7);
- Elect some staff members that will help him when the event takes place;

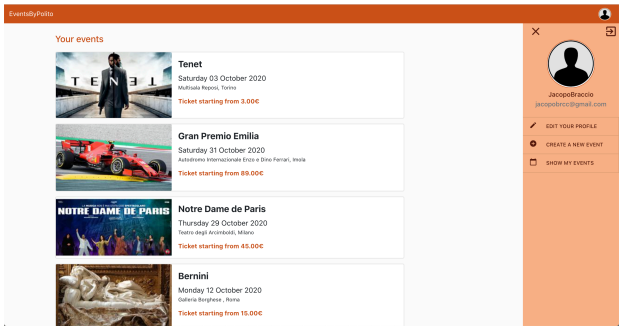


Figure 6 – Events created by provider 'JacpopoBraccio'

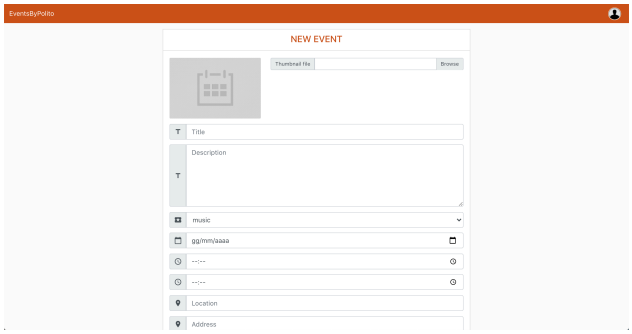


Figure 7 – Add new event page

3.3 Mobile applications

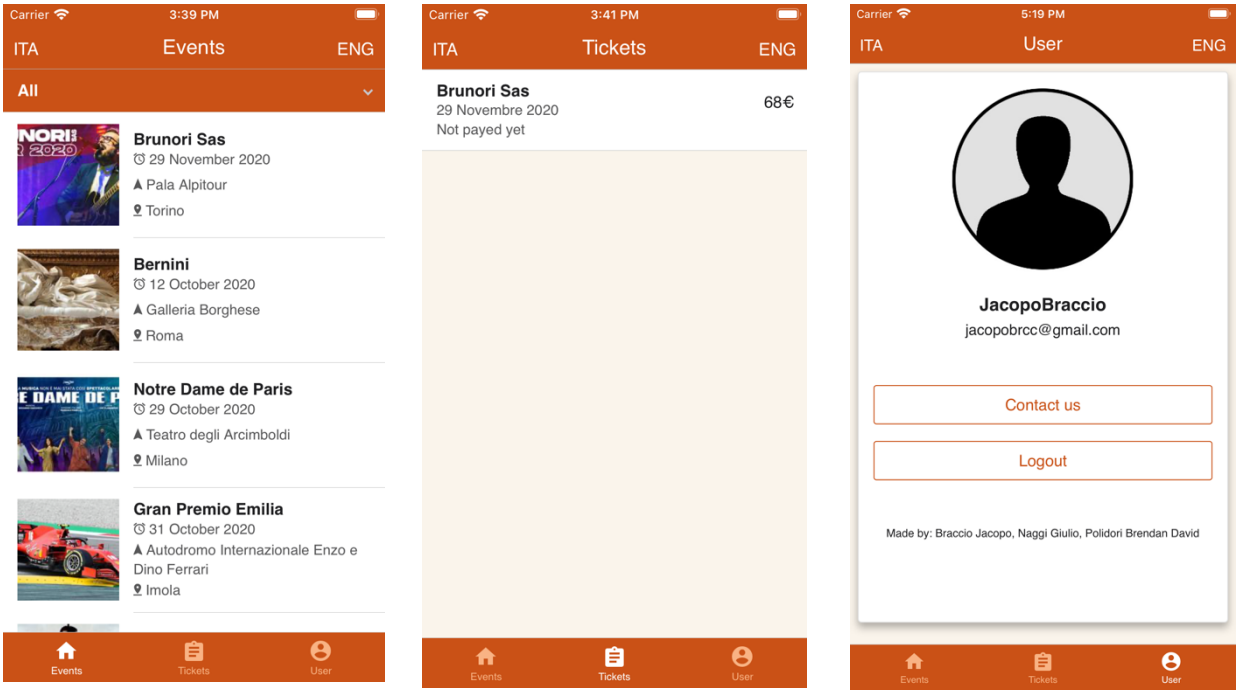
The platform also includes two mobile applications: one is completely dedicated to customers while the second is dedicated to promoters and their staff members. Both applications were developed using Apache Cordova mobile development framework. Cordova allows the use of standard web technologies (HTML5, CSS3 and JavaScript) for cross-platform development. Applications execute within wrappers targeted to each platform, and rely on standards-compliant API bindings to access each device's capabilities such as sensors, data, network status, etc. Therefore, the developed applications can be installed both on iOS and Android. The user interface was built using Onsen UI that is a large set of rich UI components specifically designed for mobile apps. Onsen UI enriches app users' mobile experience with natively designed UI elements: components change their aspects according to the platform on which they are running, so it's perfect for developing hybrid apps using Cordova or developing mobile web apps (also referred to as Progressive Web Apps).

The communication between the applications and the web-server takes place using a series of developed REST APIs and exchange of JSONs.

Italian or English can be chosen as the main language for both applications.

3.3.1 Customers' Application

The application dedicated to the customer is made of 3 main sections: Events, Tickets and User.



Events

The catalogue of all available events is shown in this section. It is possible to filter events by their type (all, music, culture, theatre, movies, sport and other).

Tickets

All the reservations placed by a user are shown in this section, together with their status (‘Payed’ or ‘Not Payed’) and their total price.

User

Section dedicated to userfrom which they can contact the system administrator or logout from the platform.

Figure 8: Main sections of customers’ application.

By clicking on one of the proposed events, a page dedicated to the chosen event opens. Figure 9, shows the event page dedicated to the event ‘Brunori Sas’. By scrolling through the page, customers have the opportunity to see some details about the event, such as its content, date and time, where it is going to take place, who has created the event and the kind of tickets available along with their available quantities. Eventually, customers can place a reservation or make a purchase using the two dedicated buttons and share it with their friends using the share button on the top right corner.

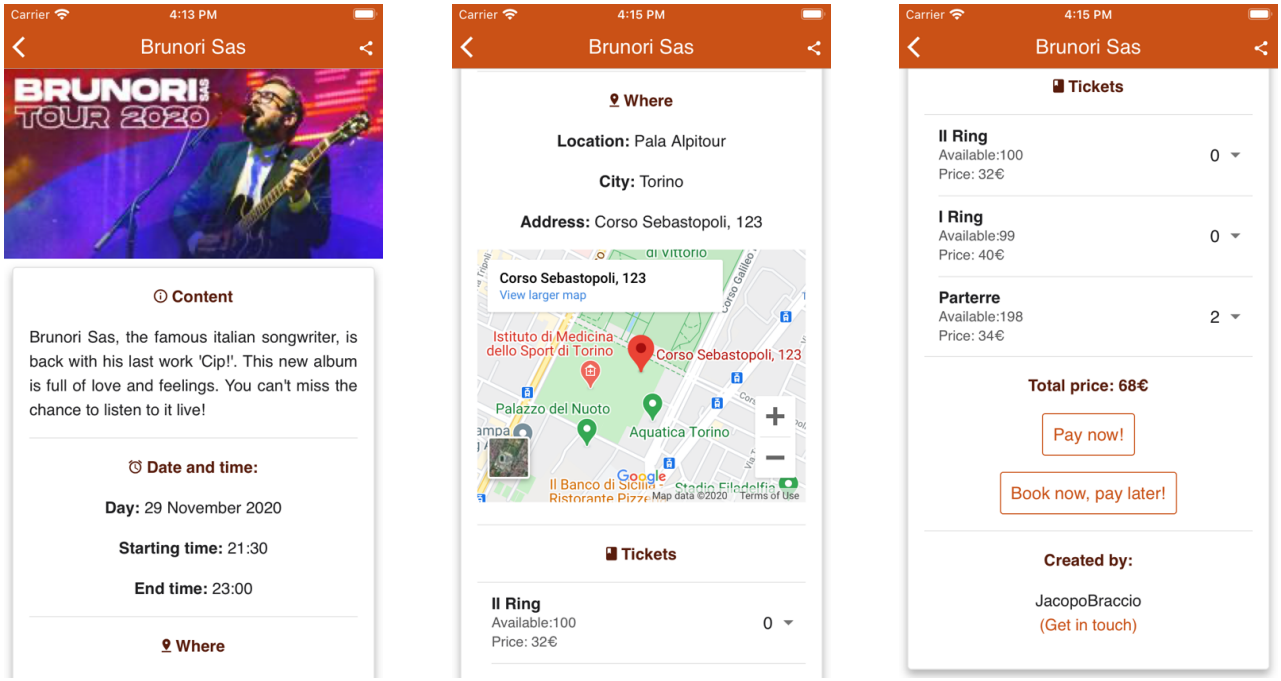


Figure 9- 'Brunori Sas' event's page.

Once a reservation has been correctly placed, a success message will be displayed, and the reservation appears under the ‘Tickets’ section. By clicking on a reservation, the page containing the tickets is opened. Figure 10 shows the 'Tickets’ section for a customer that has reserved two ‘Parterre’ tickets for the event ‘Brunori Sas’. The ticket page contains the QR code related to the reservation, some information about the event and if it has been paid or not. This page is then going to be scanned at the access gate. If different types of tickets are bought for the same event, a simple swipe to the left is going to show them.

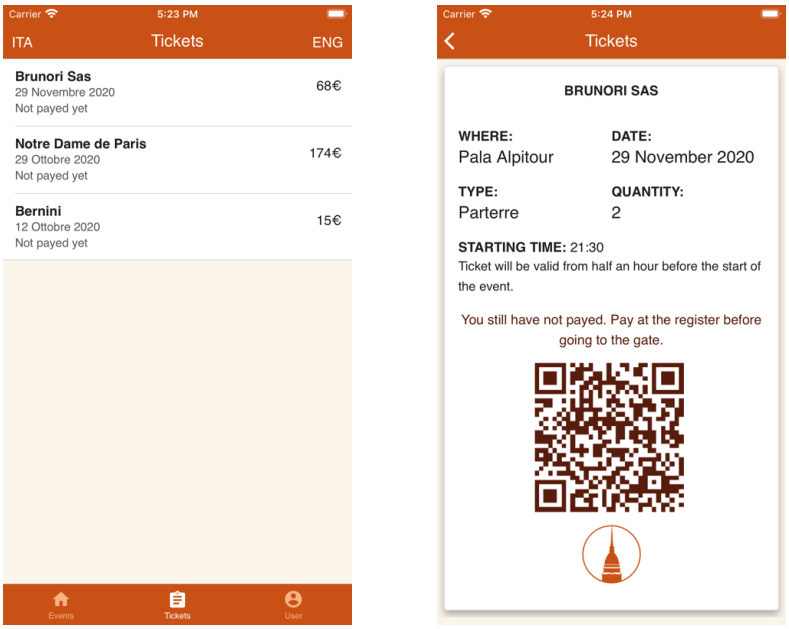


Figure 10 - Reservation of 'Brunori Sas' event.

3.3.2 Promoters' Application

Providers have the role of creating, updating and managing events, therefore an application dedicated only to these tasks has been developed, called 'EventsByPolito Promoters' that has almost the same functions of the web platform, plus some additional features.

As soon as the provider logs in the application, a main screen with all the events it has created is shown. Figure 11 represents the homepage of the provider 'Jacopo Braccio'.

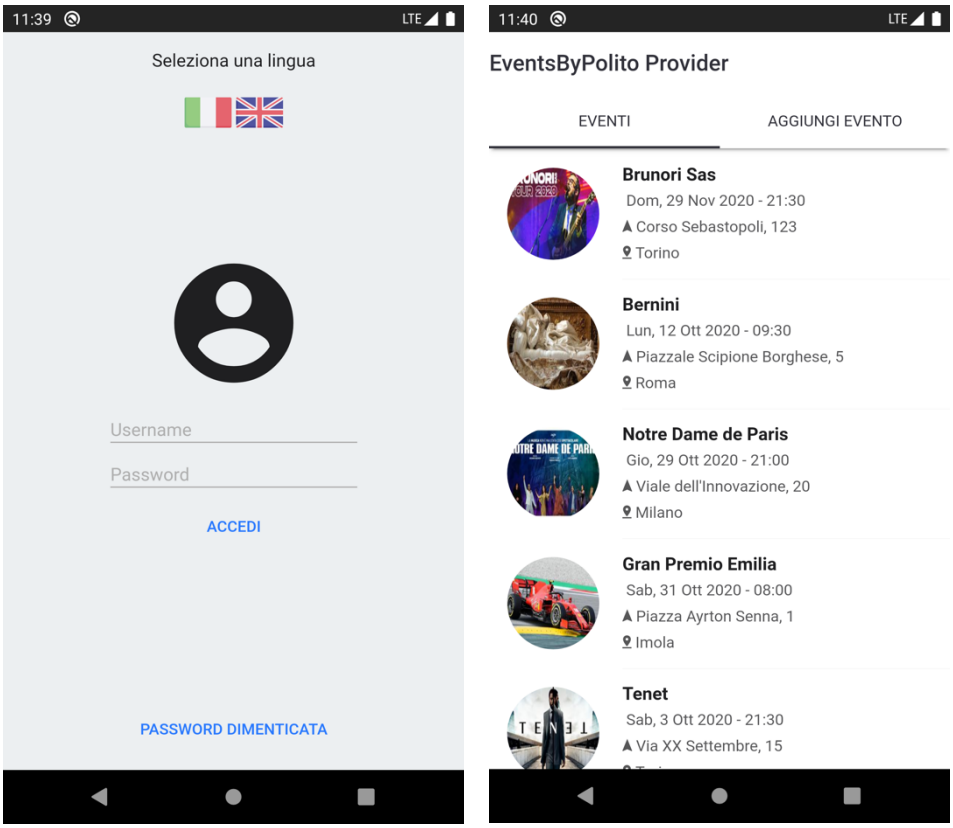


Figure11 - Login and homepage 'EventsByPolito Provider'

From the main view of the application the provider has the possibility of adding new events (by adding the same info it would add on the web platform: image, content, date and time, etc.) or modifying the existing ones, Figure 12.

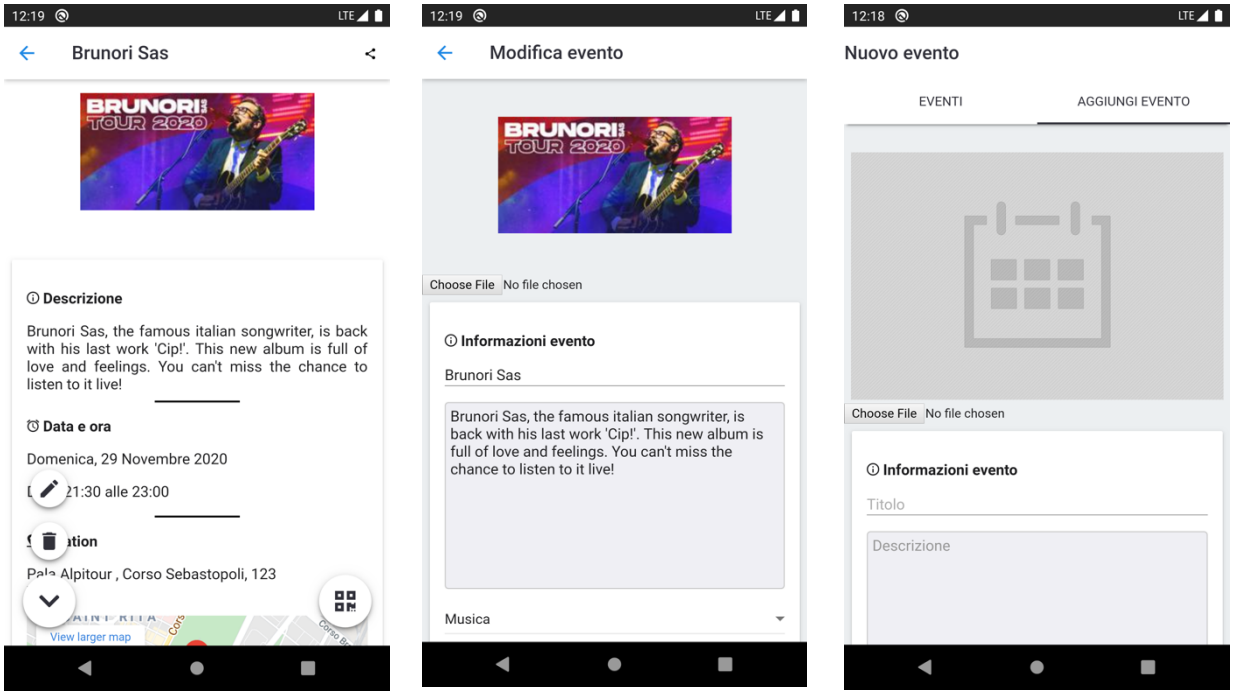


Figure 12 - 'Modify event' and 'Add event' view.

'EventsByPolito Promoters' is also accessible to members of the staff elected by those who created the event. Once the staff member is logged in, it can see the list of the events for which he has been chosen, Figure 13.

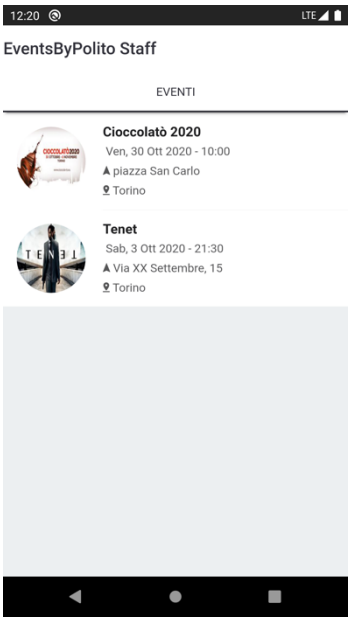


Figure 13 - Staff member 'Giulio Naggi''s homepage.

Both the staff members and the creator of the event can use the function 'Scan', accessible by a button on the event's page, implemented in the application. This function allows to scan customers' tickets and check their validity in order to let people join the event, performing a human version of the access gate (section 3.4.2), Figure 14.

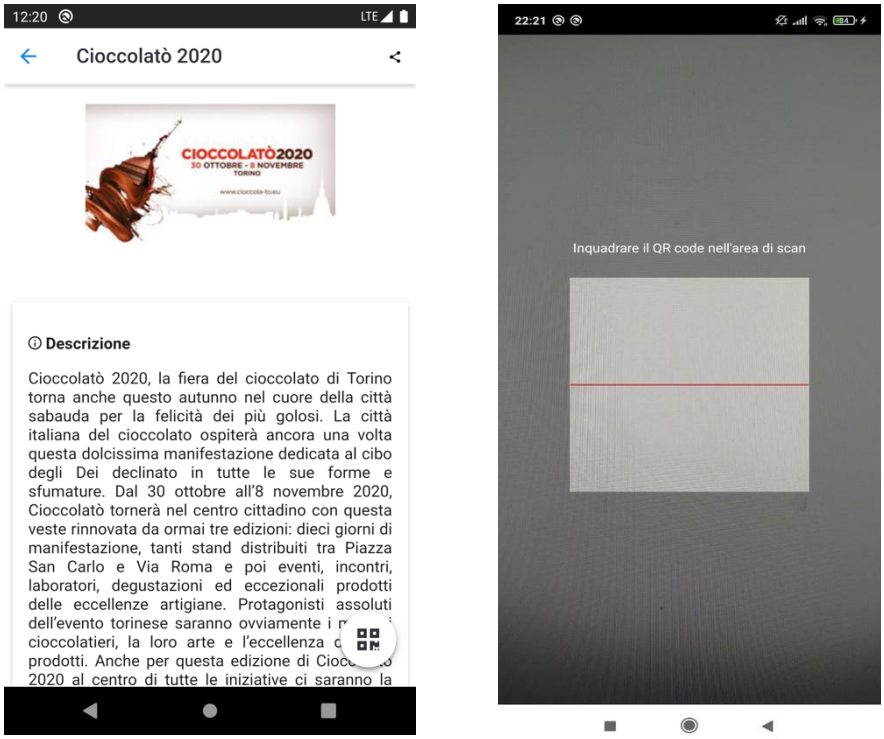


Figure 14 - Scan feature of 'EventsByPolito Provider' app.

3.4 On-site devices

Final components of the platform are totems and access gates.

3.4.1 Totem

The totem is another way of accessing the platform. It is placed in some points of interests of the city so it can be found easily by customers. Through it, customers can perform the same actions they would perform using the website: consult events, book, pay tickets and so on. The totem is based on RaspberryPi and has a 7" touch screen display. It is configured in "Kiosk Mode", that is a mode offered by browser applications (Chromium in this case) to run the application full screen without any browser user interface such as toolbars and menus. The intent of setting up "Kiosk Mode" is to prevent the user from running anything other than the browser-based content in the full screen browser window.

Thanks to the totem, people that do not have internet access (for example tourists in different countries) can access the 'Booking' section, take a picture of the QR code of their reservations, and use that picture to access the gate.

Figure 15 shows the totem and how it is set up.

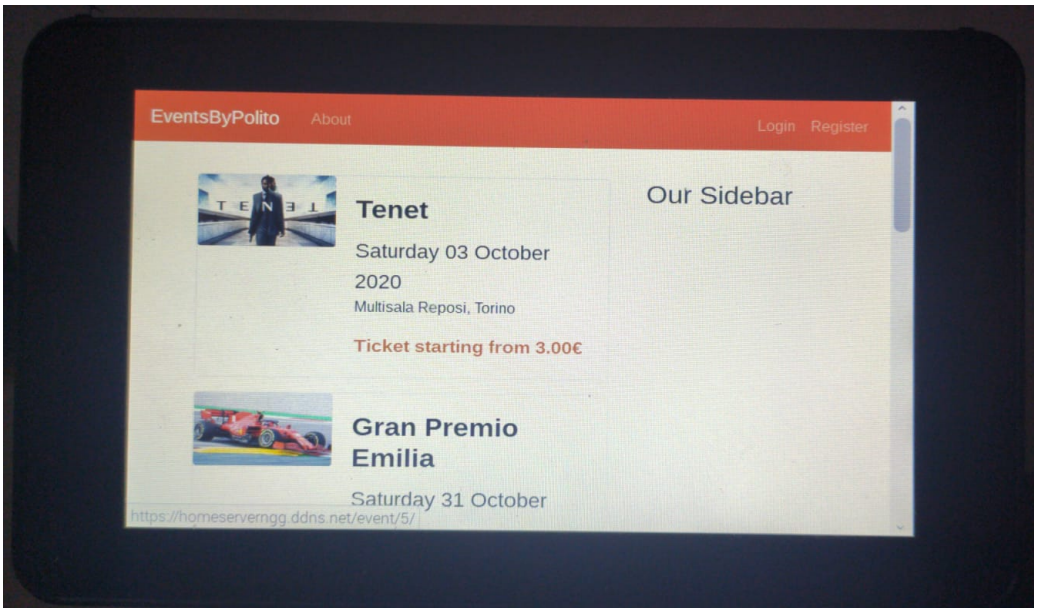


Figure 15 - Totem setup

3.4.2 Access Gate

The QR code received once the reservation is placed can be used to access the event. At the entrance of each event's location, an access gate is placed. It consists of a RaspberryPi and a Webcam through which clients can scan their code. The content of the QR is sent to the backend server that computes the validity of the ticket by checking access slot time, date of the event and if it has been paid or not. The result is shown on the access gate: if the ticket is invalid, an error message is shown, whereas if the ticket is valid, the gate opens, and the customer can access the event. If more than a ticket is bought in the same reservation, the QR code can be scanned as many times as the bought quantity. Once the total number of scannable times is reached, an error message will be shown.

Figure 16 shows the simple interface of the access gate: it only consists of two buttons used for choosing language between Italian and English and a scanning section. Totems are configured remotely and there is no form of input allowed to external entities in order to improve the security of the platform.

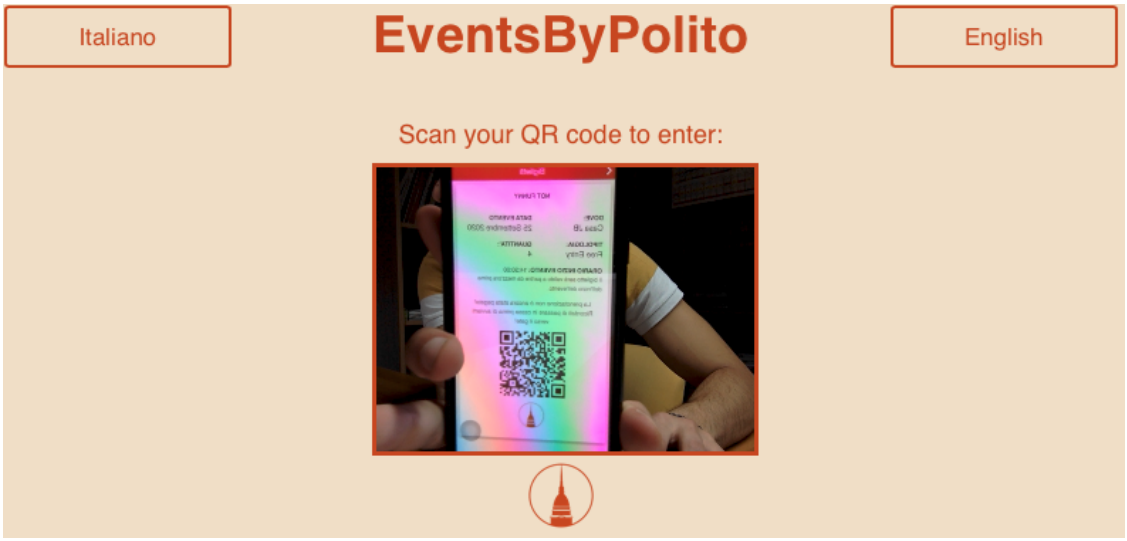


Figure X Access slot user interface.

4. Conclusion and further improvements

The platform meets the following minimum requirements asked:

- A system management interface;
- Server with database and webserver;
- Totem to place a reservation;
- Gate check.

Additional implemented features are the following:

- Client application that works both on iOS and Android;
- Management application that works both on iOS and Android;
- Reservation possible also from the web platform and the mobile app;
- Payment system integrated with 'Stripe' (right now working only on the website);
- Differentiation of actors in clients, promoters, staff members and managers;
- Possibility of accessing tickets from the application and the website;
- Possibility of receiving the tickets by email;
- Possibility of downloading specific tickets in pdf format
- Multilanguage on all the frontend parts: web-platform, applications, totem and access gate can all be set in English or Italian;
- Scan feature directly from the management app.

During our tests the designed platform always behaved as it should. All the operations were smooth and flawless. It is a prototype and of course it is designed to be used on a small scale. Some improvements could be made in order to make it even more functional and possibly

commercial. Hybrid apps, even though are easy to develop and save lots of time, are not performing as native mobile applications would. The first improvement would therefore be to translate ‘EventsByPolito’ and ‘EventsByPolito Provider’ app in iOS and Android native language (Swift/Objective-C, Java/Kotlin) in order to have a more fluid and seamless experience. Other improvements are related to clients: a section ‘For You’ could be implemented so that the platform automatically recommend events according to client’s tastes and/or his location, payments system such as PayPal or Satispay could be implemented and the option of choosing more languages should be available in order to widen the platform market.