

**Web Applications A.Y. 2022-2023**  
**Homework 1 – Server-side Design and Development**

**Master Degree in Computer Engineering**  
**Master Degree in Cybersecurity**  
**Master Degree in ICT for Internet and Multimedia**

Deadline: 28 April, 2023

<b>Group Acronym</b>	<b>ACME</b>	
<b>Last Name</b>	<b>First Name</b>	<b>Badge Number</b>
Borsato	Alessandro	2089108
Campagnol	Andrea	2091178
Cardillo	Vittorio	2091429
Lenartavicius	Vaidas	2092135
Maglie	Mattia	2095330
Marcato	Francesco	2082155
Pallante	Laura	2092566
Talukder	Md Imran Faruck	2041440
Villani	Matteo	2090299
Zago	Giovanni	2087645

# 1 Objectives

This project is intended to be a Web App that will be used by ESN Padova, an active section of ESN Italia. It would help them to streamline and facilitate tasks of registering members and maintaining a membership list different every year (currently made by volunteers). The use of a web application would then likely increase the visibility of its events to international students and make it easier to register for events organized by ESN Padua, just from their smartphones.

Furthermore, that allows volunteers to manage events and Erasmus students, since there is the opportunity to join some events only if you're a member of ESN Padova (that corresponds in getting an ESNcard and pay a fee).

At the moment, subscription is made with a paper form, the payment by cash or POS and the overall database is a spreadsheet. Events are managed by communication manager, who post them on Instagram, website and Facebook group. Mainly former platform.

As result, this is a very time consuming process, that requires a lot of effort from volunteers and, what is more, it is not very efficient.

Overall goal is to facilitate the process and build a stronger database.

## 2 Main Functionalities

Main functionalities are intended to facilitate users and event management, besides a stronger database. Above the list of functionalities asked by a group member, who's also an active volunteer in ESN Padua. After discussing them with the group, we listed these goals for the web app.

### 2.1 Database

The main idea and use of the above-presented Web Application are to provide a reliable management system to ESN Padova.

In fact, at the moment to manage all the ESN Padova members the association uses a Google Spreadsheet, where they're inserted through a Google Form.

This procedure, besides not being reliable and not very agile to changes, may be subjected to input errors and several works every time some specific list (i.e. participant list to a specific event).

A centralized DB (i.e. PostgreSQL) and a user-friendly interface where Erasmus students and International people can autonomously become members can reach the goal of slimming down the work done by the volunteers to manage and keep this spreadsheet updated.

As a second but welcome goal, ESN Padova is also looking for a place where members can see and pay for events. In fact, at the moment the main place where members get to know about an event is through social networks or the website, where it may not be easy to navigate given the amount of information on the website.

### 2.2 User Management

Erasmus and International students should be able to register and create a new account autonomously through the Web Application. Once they confirmed their email, they are assigned a tier 0 user.

Tier 0 users have limited users and they can only see and participate in certain events that are open and available for everyone and don't require any kind of registration.

Tier 0 users are not to be considered members since they don't possess and paid for an ESNcard (association card), and they didn't insert all the information necessary to be considered a member.

A Tier 0 has to possibility to become a Tier 1 user and become a member of the section once he/she fills in the subscription form and pay the card fee.

Once they did what is above, they can come to the ESN-office with a precompiled document that needs a signature, and they will receive their ESNcard.

Tier 1 users are, in a nutshell, all the Erasmus and International people that are a member of ESN Padova and can participate in all the event available for them.

Tier 1 users can indeed see all the events, participate and pay the fee(if needed) through the use of the application.

## 2.3 Event Management

The Web Application should have an Event section where Erasmus and International students can easily navigate and see what are the next available events, see the details and participate if they want to.

To create, manage, and easily consult events that ESN Padova organizes, we need to define also other two types of Users, in particular, Tier 2 and 3 users.

Tier 2 users are active ESN volunteers, and as one they can create, and manage their events and provide all the necessary information.

Tier 3 users are admin users that can do all the above with no restriction on visibility.

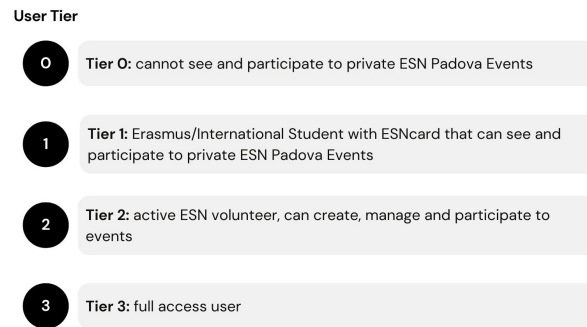


Figure 1: Users tiers.

### 3 Data Logic Layer

#### 3.1 Entity-Relationship Schema

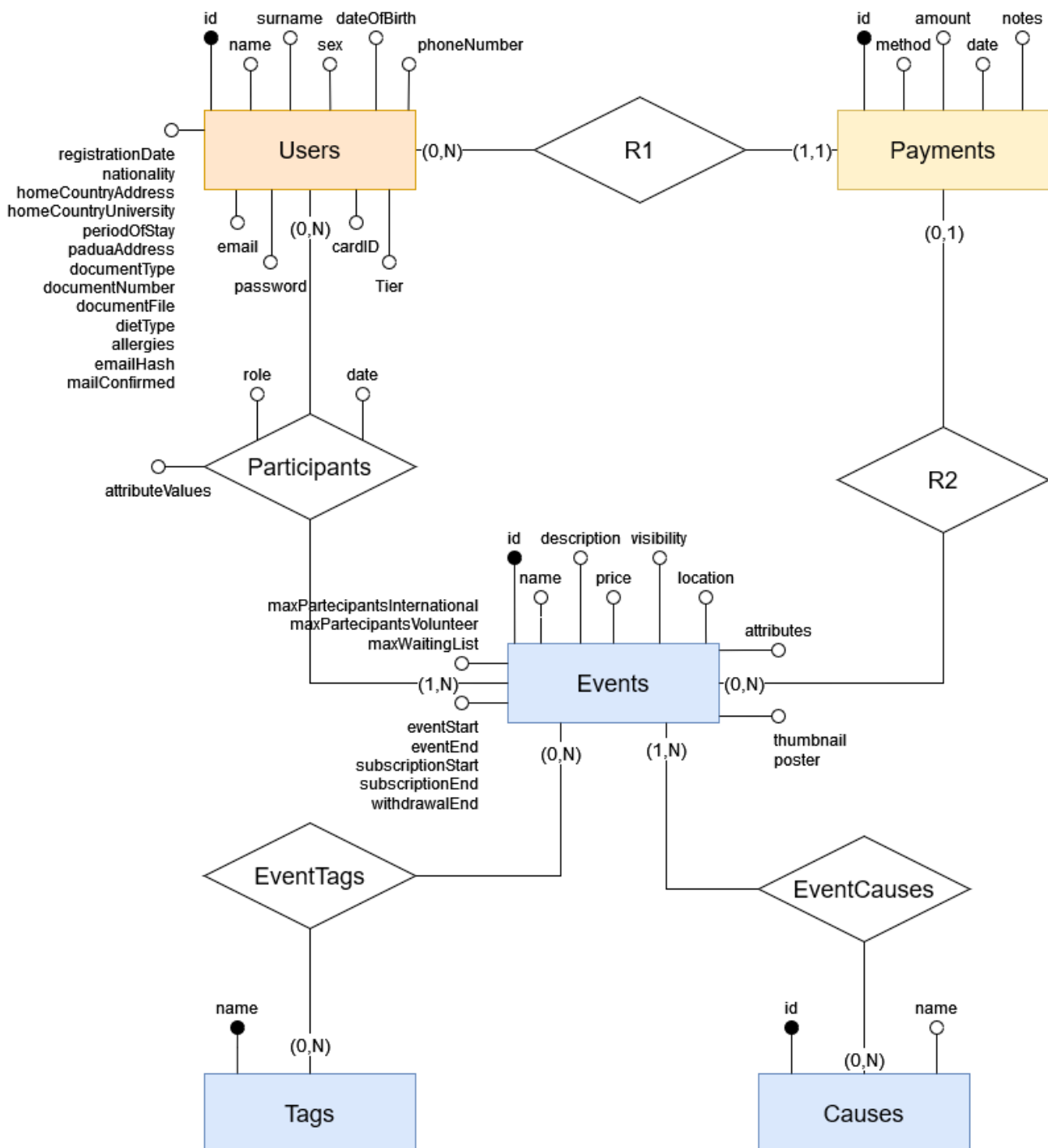


Figure 2: Database ER schema

Our database has 3 main entities, plus an important relation. Others are auxiliary and will be explained shortly at the end. Except only one entity, all entities have an integer ID as primary key. Main entities are:

- **Users:** this entity contains all the information about every kind of users, from tier 0 to system admin. This table has new entries every time a registration form is filled: the dummy form will generate a tier 0 user, while the real form will generate a tier 1 user. Every attribute is self-explanatory and stored in a very intuitive way: name and surname are varchar(50), all dates are type date with local time zone and so on. Here, just as examples, some attributes with their SQL code and few information about them.

```
CREATE TYPE diet AS ENUM ( 'NoSpecific', 'Vegetarian',
                           'Vegan', 'Halal', 'Kosher', 'Pescatarian' );
```

```
CREATE TABLE public."Users" (
  id SERIAL PRIMARY KEY,
  email VARCHAR(255) NOT NULL UNIQUE CHECK
    email ~* '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' ,
  ...
  "homeCountryAddress" json ,
  ...
  "documentFile" text ,
  "dietType" diet ,
)
```

In the above snippet, we can see few special features: RegExpr is used to check mail spelling, addresses and similar variables are stored as json, for an easier data access. Diet and others are stored as enum, to avoid user typo and having more readable attributes.

- **Events:** as for users entity, event has a long list of attributes, wisely chosen in relation how events are actually organized. Some events are restricted to certain users, based on tiers, or have a limited number of participants. All these particularities are fully represented in the entity.
- **Payment:** this entity contains all the informations about payment for event participation. Of course the actual web application isn't linked to real payment methods, but once it will be, this Entity will be related to real payments, in order to tracing them, know most favorite payment methods etc.
- **Participants:** opposed to previous entities, this is a relation between users and events, collecting user role, as listed in this snippet:

```
CREATE TYPE roleTypes AS ENUM ( 'Organizer', 'Participant',
                                'Volunteer', 'WaitingList' );
```

Furthermore, date helps managing waiting list and attributes

## 3.2 Other Information

Here some informations about minor entities and relations.

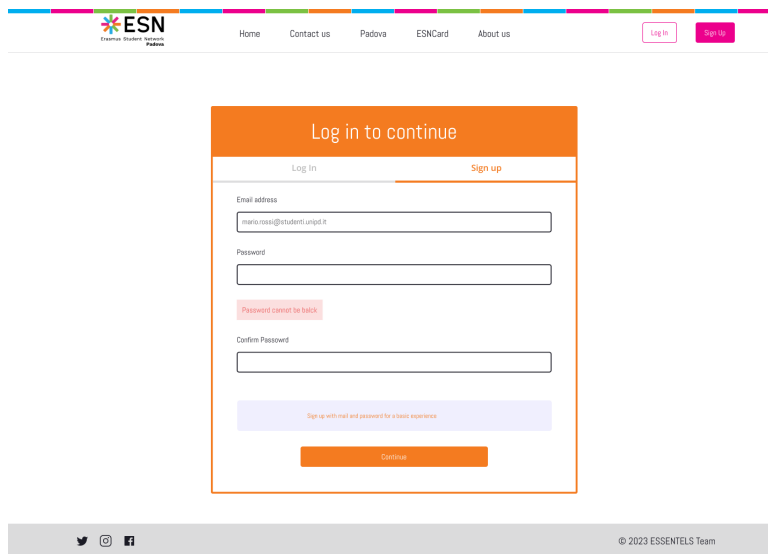
**Tags** and **Causes** are two entities that are used to classify events. Every ESN event supports at least one cause, such as and can be tagged as . These two entities are used to filter events, in order to help users to find what fits most their interests.

**EventTag**, **EventCauses**, **R1** and **R2** are relations without attributes, used only to join tables.

## 4 Presentation Logic Layer

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

### 4.1 Signup page



The screenshot displays the 'Sign up' page of the ESNTL Student Services Padova website. The page features a colorful header with the ESNTL logo and navigation links: Home, Contact us, Padova, ESNTL Card, and About us. A 'Log in' button is also present in the header. The main content area is titled 'Log in to continue' and contains a 'Sign up' form. The form includes fields for 'Email address' (with the example 'maria.rossi@studenti.unipd.it'), 'Password', and 'Confirm Password'. A red error message 'Password cannot be blank' is visible below the password field. Below the form, there is a blue button labeled 'Continue' and a link that says 'Sign up with email and password for a basic experience'. The footer contains social media icons for Twitter, Instagram, and Facebook, along with the copyright notice '© 2023 ESSENTLS Team'.

Figure 3: Sign up page

Users can sign up just with mail and password. They will not have access only to their 0 events. For a better experience and more opportunities they are asked to complete their profile with a more detailed form, including these from their hometown. This form is shown in 4 and described in next section.

## 4.2 Registration form

The screenshot shows the ESN website header with a colorful bar and navigation links: Home, Contact us, Padova, ESNCard, About us. There are 'Log in' and 'Sign up' buttons. The main content area is titled 'Log in to continue' and has two tabs: 'Log in' and 'Sign up'. The 'Sign up' tab is active. The form contains the following fields: Full Name (filled with 'Mario Rossi'), Home country university (filled with 'Harvard'), Home Country Address (filled with '7th street, New York'), Padua Address (filled with 'Via Roma 12, Padova (PD)'), Phone number (filled with '+39 340 7777 555'), and Date of birth (filled with '01/08/1984'). A 'Sign up' button is at the bottom of the form. The footer includes social media icons and the copyright notice '© 2023 ESSENTELS Team'.

Figure 4: Form for the full registration

When user wants to upgrade to tier 1 they have to pay their subscription (as shown in 5). The form for a full profile asks for many information, here just a few are displayed, but the real form will have more than 10 input boxes.

## 4.3 Payment page

The screenshot shows the ESN website header with a colorful bar and navigation links: Home, Contact us, Padova, ESNCard, About us. There is a user profile icon. The main content area is titled 'Payment Method' and has a back arrow. It contains two sections: 'Chose a payment method' with radio buttons for 'Cash' and 'Credit card', and 'User details' which displays the user's information: Full Name (Mario Rossi), Email (mail@gmail.com), Home country address (7th street, New York), Home country university (Harvard), and Padua Address (Via Roma 12, Padova (PD)). A 'Pay Now' button is at the bottom of the user details section. The footer includes social media icons and the copyright notice '© 2023 ESSENTELS Team'.

Figure 5: Payment page

When a user pays his subscription, he will become a tier 1 user. He will be able to participate to more events. Still he cannot create an event (only tier 3 or above users can do that).

## 4.4 Events list page

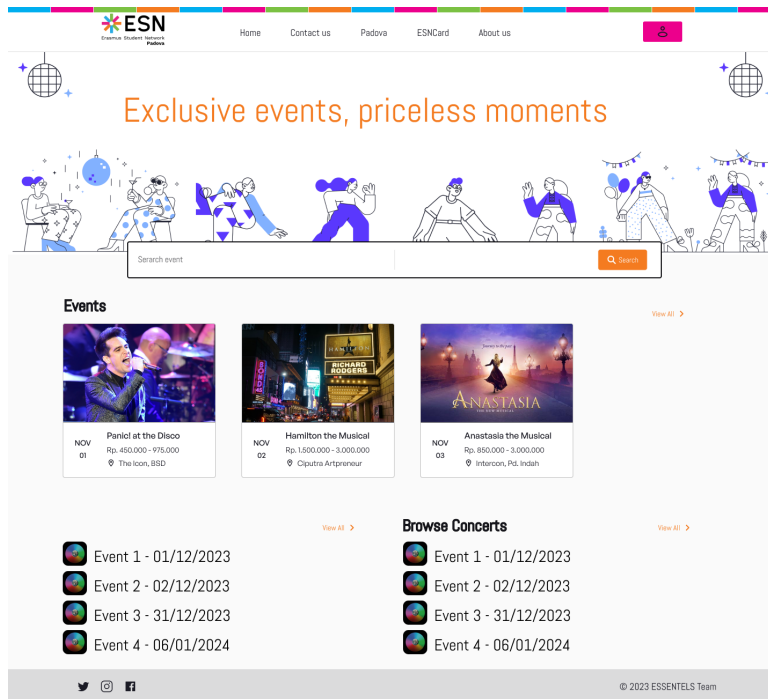


Figure 6: Sign up page

Here all events are listed or filtered by tag. Users are allowed to see only events for their tier or lower ones. Since all events have a location, we decided -according to ESN volunteers- to keep events automatically filtered by tier. Searching by tag or by cause are actually made by a REST API: using GET events are listed, administrator can also use DELETE to delete an event. With a POST request, a new event can be created.



## 4.5 Join event page

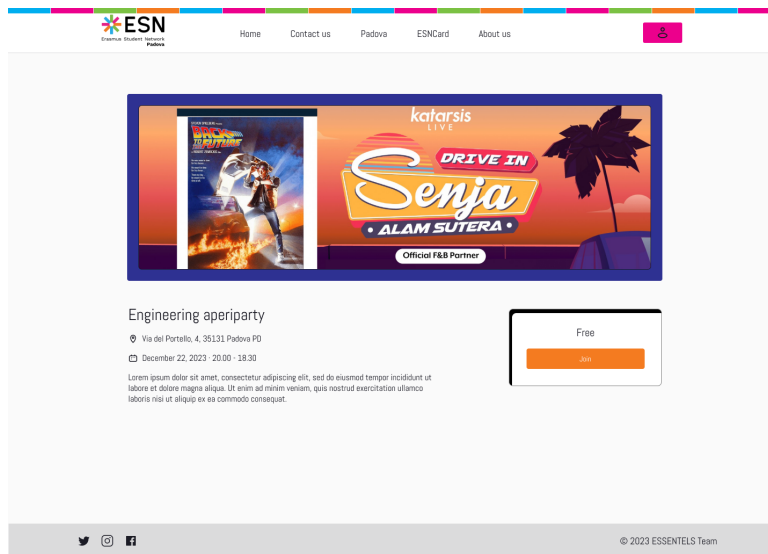


Figure 7: Sign up page

When a user wants to join an event, he can see all the details and the list of participants. Here he can also see more detailed info about the event and confirm his participation. In case some events will ask for a ticket one day, this page will redirect to a payment page, similar to 5, but linked to tickets for cinema, party etc.

## 4.6 Change user info

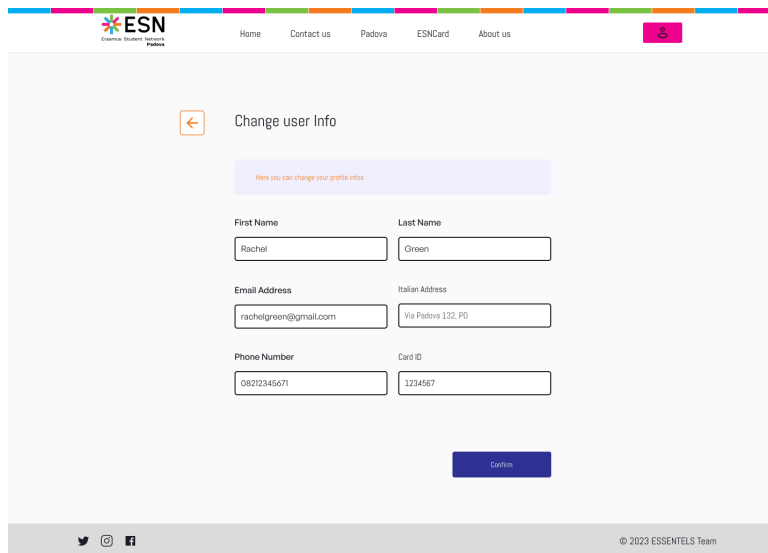


Figure 8: Change user's info

Users can change their profile info. This page can be accessed by each user for his own profile or by an admin for any user.

## 5 Business Logic Layer

### 5.1 Class Diagram

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

### 5.2 Sequence Diagram

Here reported the sequence diagram for the event creation operation. To create a new event, an Admin user (at least tier 3), fills the creation form in the Create Event page. By clicking on Continue button, a POST request is issued to the web server, specifying the URI: /create-event.

The data passed to the web server are name, description, price, visibility, place (city, street and number), maximum number of international participants, maximum number of volunteer participants, start date, end date, start subscription date, end subscription date, withdrawal end date, size of waiting list, attributes, thumbnail image and poster image.

The web server instantiates the CreateEventServlet which calls its doPost method, passing the HttpServletRequest and HttpServletResponse. This method handles some POST data packing city, street and number in a unique location JSON Object and storing the two images (thumbnail and poster) in the "ESSENTLS\_Cloud" folder. A new Event object is created, containing the other POST data, the generated location JSON object and the paths that represent the images. The control is passed to the AdminCreateEventDAO which receives as arguments the connection (defined in the AbstractDatabaseServlet extended by the CreateEventServlet) and the instantiated Event. The AdminCreateEventDAO (that extends the AbstractDAO ) contacts the Database Server with the access() method that calls the doAccess() override method in the AdminCreateEventDAO, which executes the SQL statement for event creation. The operation ends returning a message that informs the user that the event is successfully created.

### 5.3 REST API Summary

URI	Method	Description	Filter
What is the URI?	GET	It returns information about a user searched by id	is it behind a filter?
What is the URI?	GET	It returns a list of events with a certain tag	is it behind a filter?
What is the URI?	POST	It modifies event information	is it behind a filter?
What is the URI?	PUT	It create smt (TODO:)	is it behind a filter?
What is the URI?	DELETE	It delete an event by its id	is it behind a filter?

Table 2: Describe in this table your REST API

## 5.4 REST Error Codes

Error Code	HTTP Status Code	Description
Error Code Identifier	Corresponding HTTP Status Code	Description of the error

Table 3: Describe in this table your REST API error codes

## 5.5 REST API Details

### A resource

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

- URL: the URL to retrieve it
- Method: Method to retrieve it
- URL Parameters:
- Data Parameters:
- Success Response:
- Error Response:

## 6 Group Members Contribution

**Everyone** has contributed in the project by sharing ideas, suggestions and fixing issues in a proactive way. We all took part in the analysis process and decided to split the development of some parts of the project equally (DAOs, Servlets, jsp, ...).

**Alessandro Borsato** has developed part of the registration process, proposed a common approach to manage the project (ticketing and git) and wrote internal guides to let everyone aligned.

**Andrea Campagnol** has contributed to this project by taking care of some initial setups, developing the login and mail verification parts. He also draw the main schemas present in this document.

**Vittorio Cardillo** has helped the team in the initial setup and developed some key parts like the homepage.

**Vaidas Lenartavicius** has developed user edit page and others parts of this project. He was always available support teammates in case of fixing bug and problems.

**Mattia Maglie** took care of the payment section of the app and did also developed some filters. Also he suggested some useful refactoring of the code and best practices.

**Francesco Marcato** has proposed the idea for this application, as a result of a practical need in Padua ESN management. He wrote the requirements of this application and wrote code mainly for the events part.

**Laura Pallante** has contributed to the project by implementing some key pages and structuring REST architecture on some of those. She was also helpful in case of fixing some issues.

**Md Imran Faruck Talukder** has contributed to the project by taking care of the database implementation, also helping teammates setting up the connection between the application and db.

**Matteo Villani** has developed some filter parts and edit pages. He took care of aligning the style of the app pages.

**Giovanni Zago** has set up the Trello application we used to divide the work among us, also providing a simple tutorial on how to use it properly. He took care of writing most of the parts in the documentation (included pages mockups) and also some application pages, s.e. editing users information as user or as administrator.