800 1222·2022 ANNI
UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Web Applications A.Y. 2023-2024

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: 29 April, 2024

Group Acronym	WA001	
Last Name	**First Name**	**Badge Number**
Antonutti	Manuel	2130332
Carlesso	Daniel	2088626
Frigione	Luigi	2060685
Shams	Mahshid	2122316
Ursino	Nicola	2119984

1 Objectives

The primary objective of our web application, named PL8S, is to simplify the food ordering process at festivals by providing a user-friendly platform that eliminates the need for long queues at the counter. This web service is designed to let users autonomously compile their order, similar to the touch-screen ordering systems found in popular fast-food chains, but with the added convenience of being accessible directly from the user's device.

The application allows customers to place orders at multiple restaurants through a unified interface that can be easily accessed from home or on-site, thereby making the food ordering process more efficient and accessible.

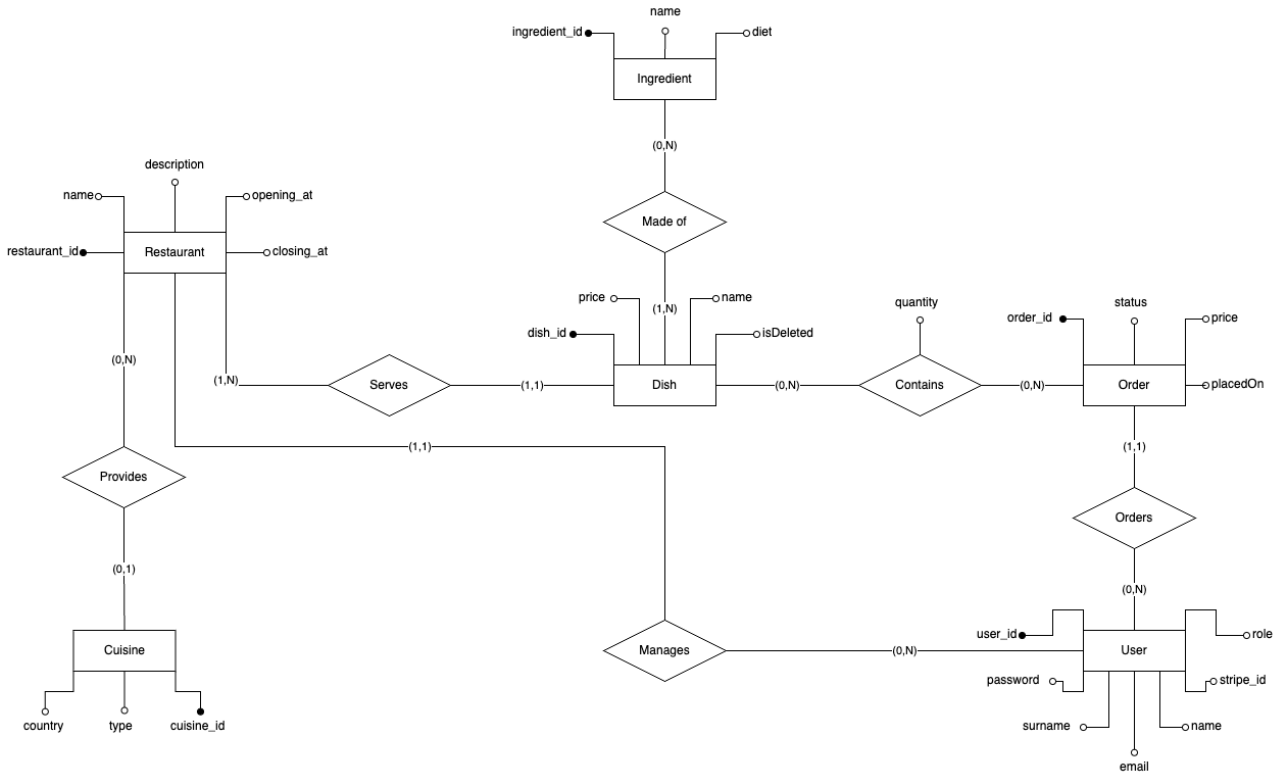
2 Main Functionalities

The PL8S web application is designed to provide a way of ordering food from various restaurants at festivals. Its core functionalities include:

- Menu Display: The application displays the menus from each restaurant, allowing users to browse through different dishes and their ingredients. This feature is particularly useful for users with specific dietary requirements or allergies, enabling them to make informed choices before placing an order.
- Advanced Search Tool: To enhance the user experience, PL8S offers an advanced search tool that enables users to find specific dishes across all participating restaurants. In this way, the customers can quickly find their favourite dish.
- Virtual Shopping Cart: Once a user selects a dish, it is added to a virtual shopping cart. This cart allows users to adjust the quantity of each item, remove selected dishes, and view the total bill. This functionality ensures a smooth and efficient ordering process, providing users with full control over their order before finalizing it.
- User Authentication: Upon launching the web application, users are presented with a login and registration page. This feature allows new users to create an account with a username and a password, which can be modified later. Unregistered or unlogged users are considered as customers and can browse the Customer View without the ability to manage a cart or place orders.

3 Data Logic Layer

3.1 Entity-Relationship Schema



We have 6 entities in the schema:

- **Restaurant**: each restaurant is defined by *restaurant_id* which is of type SERIAL. Then we have *name* and *description* of type VARCHAR, in conclusion we have *opening_at* and *closing_at* which are of type TIME and describe the opening and the closing hour respectively.
- **Cuisine**: it expresses the type of cuisine that a restaurant provides. It is defined by *cuisine_id* which is of type SERIAL. Then *type* which is of type VARCHAR and expresses the type of food the cuisine provides. In conclusion, *country* whose value is selected between a finite set of 239 countries inside an ENUM type.
- **Dish**: each dish is defined by *dish_id* which is of type SERIAL. Then we have *price* which is of type REAL, *name* which is of type VARCHAR and *isDeleted* which is of type BIT and represents if a dish has been deleted or not, the dish cannot be literally deleted by the database otherwise orders' history consistency would not hold.
- **Ingredient**: each ingredient is defined by *ingredient_id* which is of type SERIAL. Then *name* is of type VARCHAR, *diet* whose value is selected between a finite set of 3 values ("vegan", "vegetarian", "carnivorous") inside an ENUM type.
- **Order**: each order is defined by *order_id* of type SERIAL. Then *price* of type REAL which is the weighted sum of the prices of the dishes in the order, where the weights are the quantities of each dish, *placedOn*, of type TIMESTAMP, defines the day and time at which an order has been placed, then *status* whose value is selected between a finite set of 2 elements ("pending", "completed") inside an ENUM type.
- **User**: each user is defined by *user_id* which is of type SERIAL. Then *email* is of type VARCHAR and it's value is unique within the table, *name* and *surname* which are of type VARCHAR, *stripe_id* which is of

type VARCHAR and it is necessary to perform a payment, *role* which is of type VARCHAR and takes value between a finite set of 3 elements ("customer", "manager", "admin") inside an ENUM type.

3.2 Other Information

All the entities id's have been chosen to be of type SERIAL for performance reasons, since a JOIN clause performed with an INT value is faster than when performed with a VARCHAR value.
The Dish's attribute *price* must be strictly positive, a direct consequence is that also Order's *price* must be strictly positive.
The Dish's attribute *isDeleted* is set to 0 by default. The dishes can be only softly deleted for guaranteeing an order history to the customer. Instead, if we delete a user, the deletion of the associated orders is guaranteed to avoid foreign key errors.
The *quantity* attribute in the relation between Dish and Order expresses the quantity of each dish in an order, its value must be a strictly positive integer, the default value is set to 1.
The *stripe_id* in the user entity is generated within the Stripe's context and it is necessary for performing a payment.

4 Presentation Logic Layer

The web-app is designed to be partially explored even if the user is not logged in, in particular, "guest" users can still explore the restaurants list and their menu.

4.1 User View

The home page, figure 1, allows the user to browse through all the available restaurants where it is also possible to search for a restaurant by name (part of the name) and by cuisine type. Once clicked on a restaurant from the list, a new page is shown with all the dishes a restaurant can sell (figure 2). Here we can also notice how the side panel changes if the user has already logged in or not. If a guest user tries to add an item to the cart a login popup will show (figure 3). In figure 4 and in figure 5 we can see respectively the user registration and the user update forms.

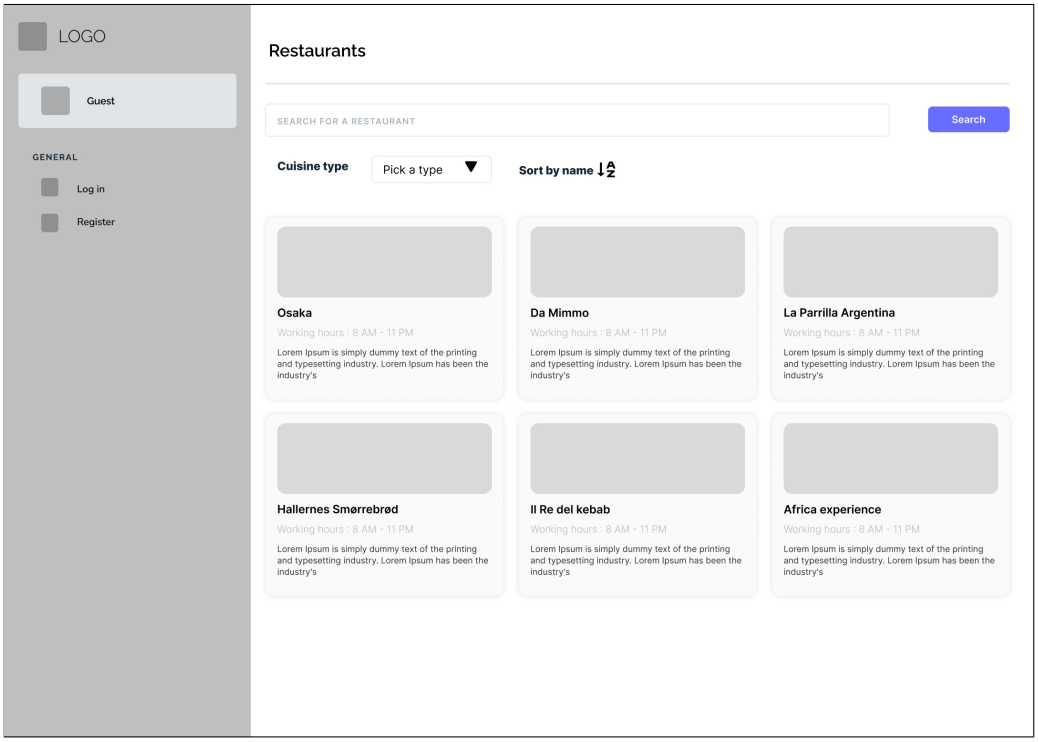


Figure 1: Home Page.

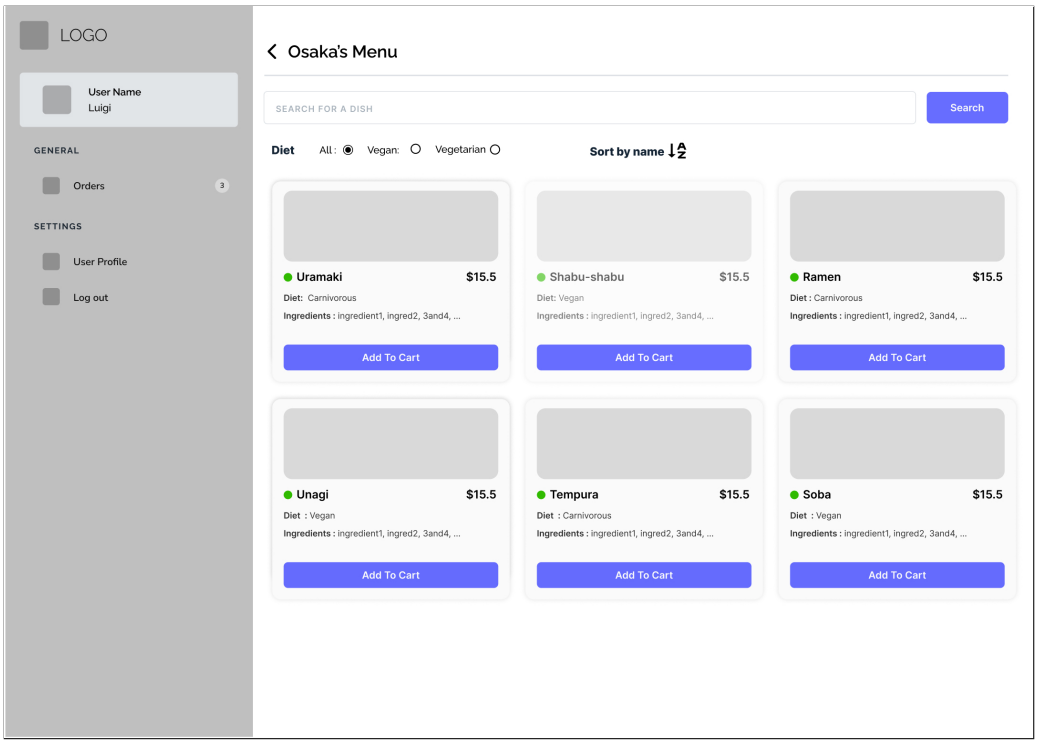


Figure 2: Restaurant's menu page.

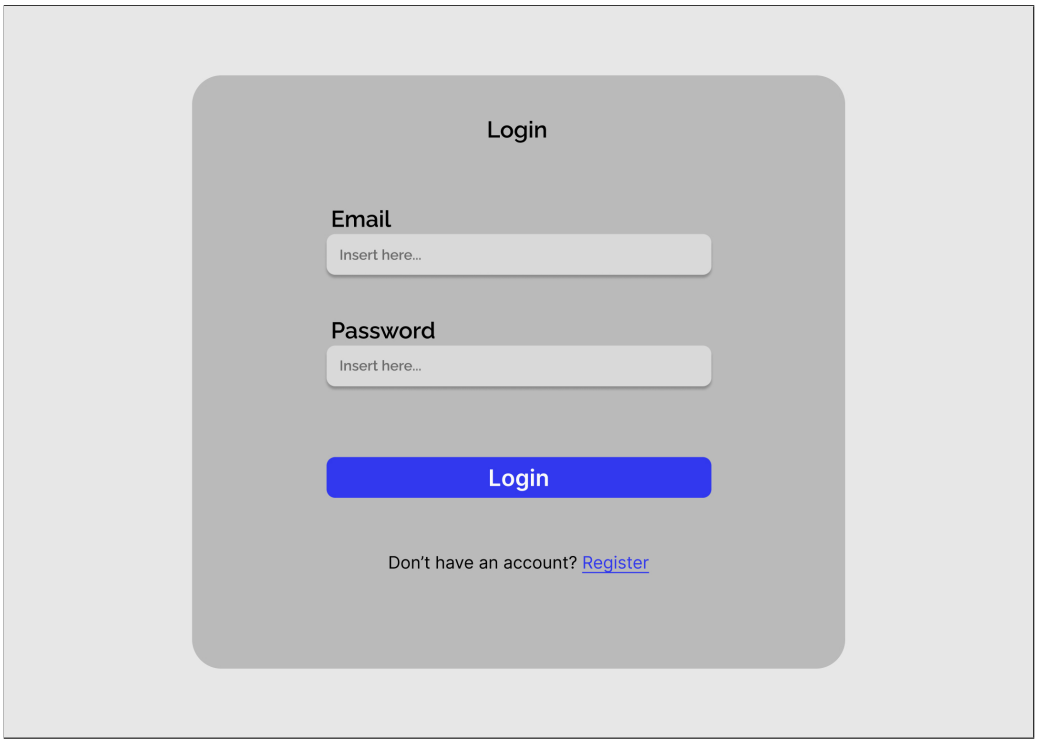


Figure 3: User login page

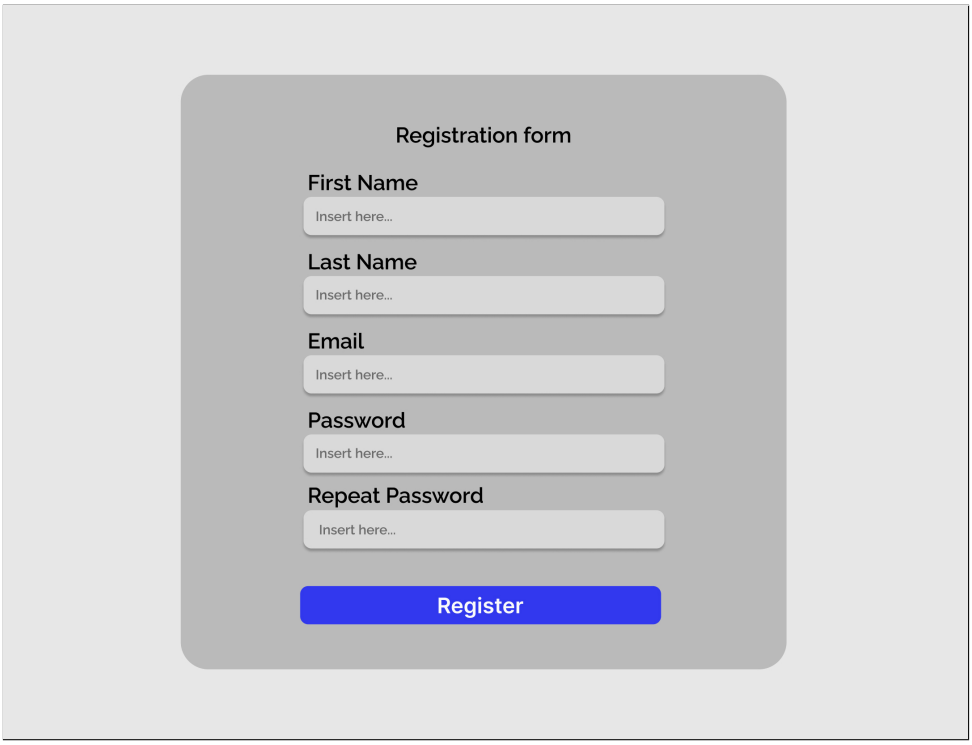


Figure 4: User registration page.

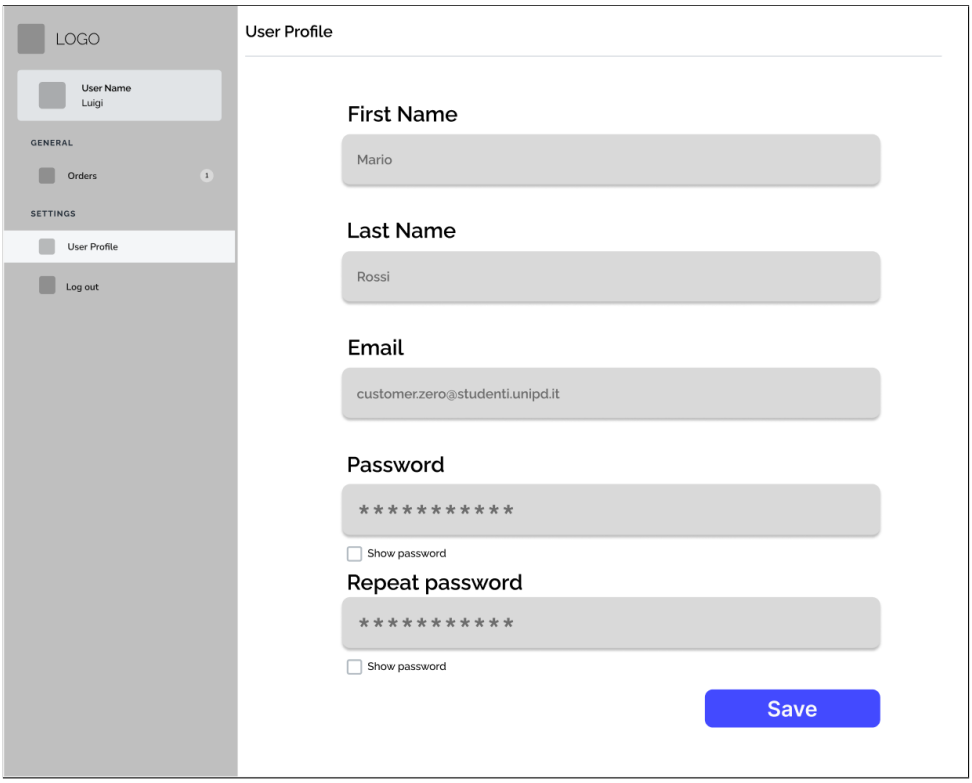


Figure 5: Update user profile page.

4.2 Admin View

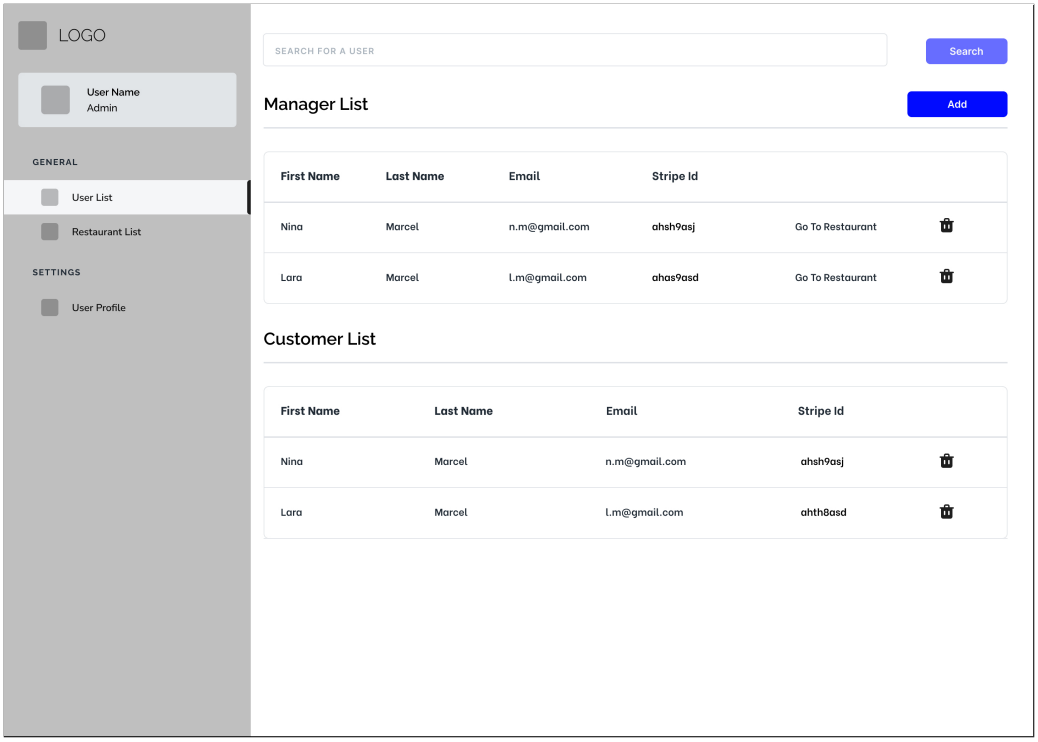


Figure 6: List of users from Admin view.

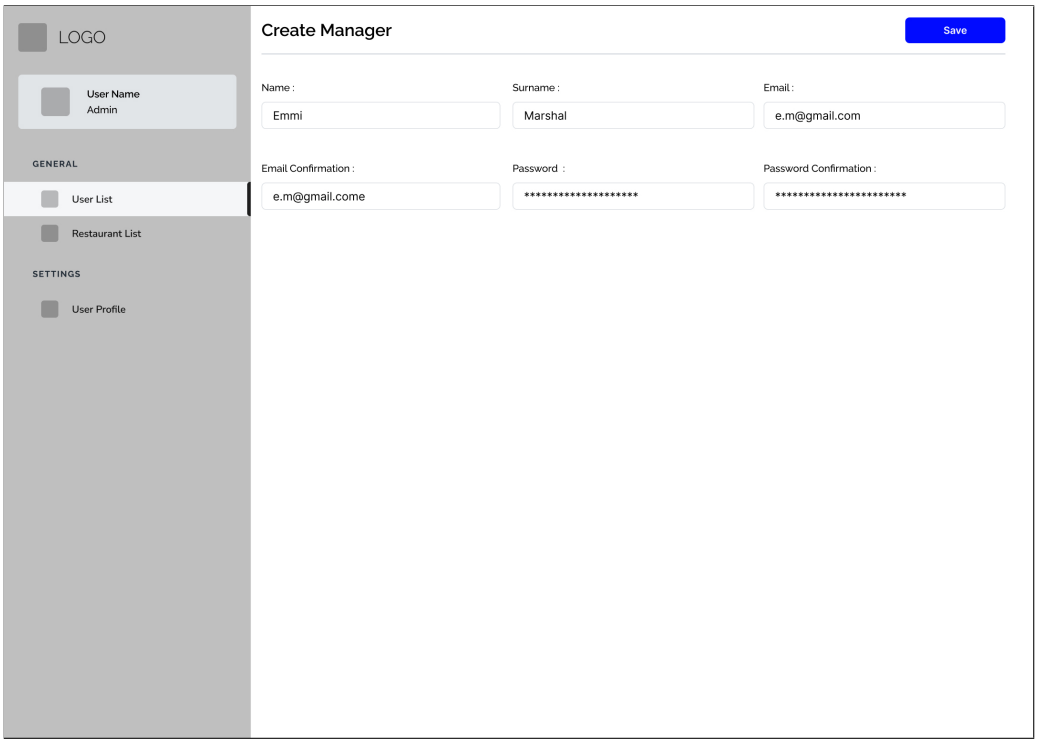


Figure 7: Admin page for creating a new manager.

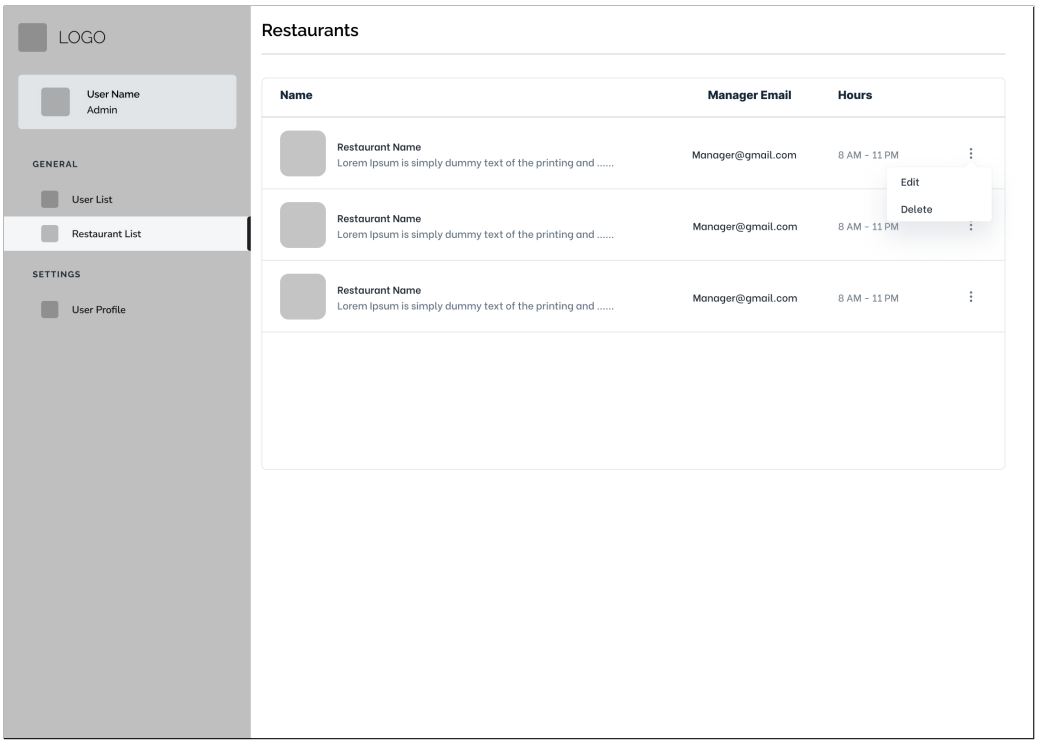


Figure 8: List of restaurants from Admin view.

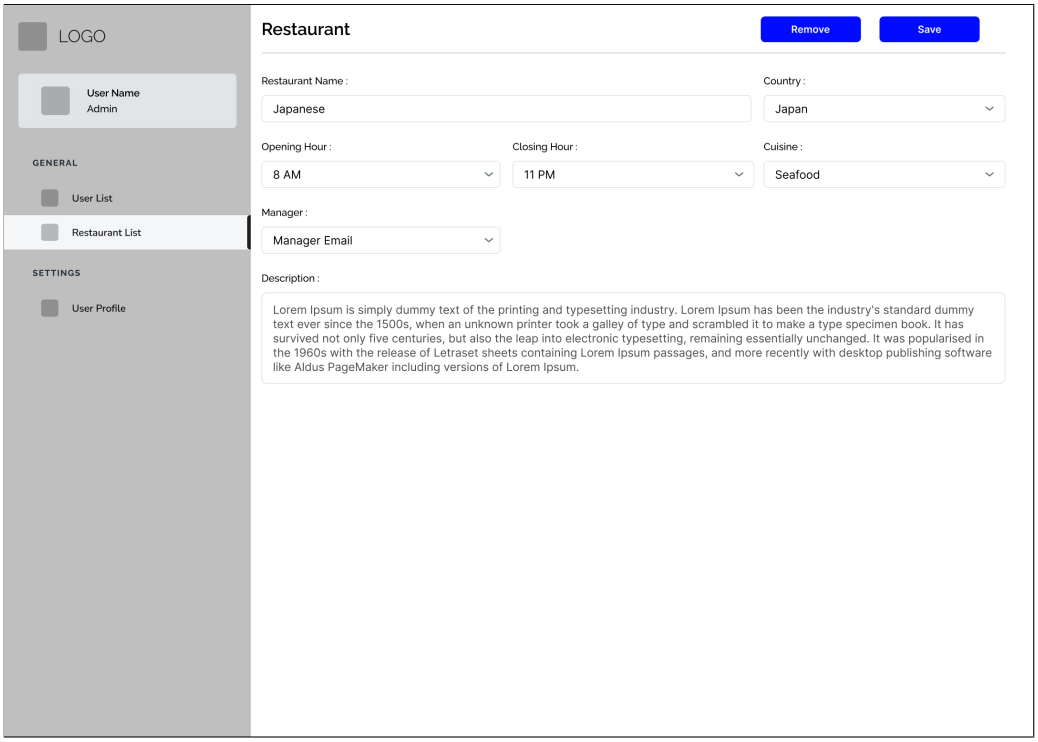


Figure 9: Restaurant editing page from Admin view.

The list of users, Figure 6, can be seen only by the admin, who can remove users or add a new manager. We

set a specific page for creating a new manager, Figure 7, accessible only by the admin, because, prior to creation, the admin and the new manager should have met and discussed about the possibility of bringing the manager's restaurant(s) to the festival. The admin can also see the list of all restaurants or the one(s) associated to a specific manager, Figure 8, and can also edit the restaurant info, because at some point a restaurant could be changing the manager and the admin is the one that can edit the manager field on each restaurant, as shown in Figure 9.

4.3 Manager View

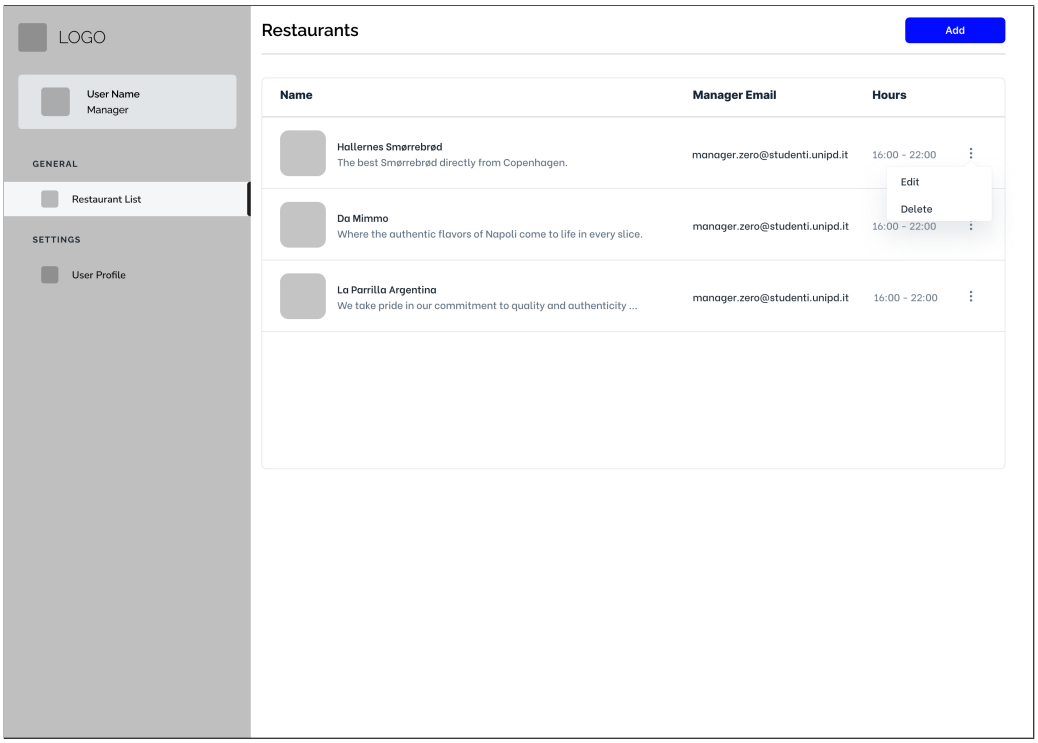


Figure 10: List of restaurants from manager view.

For the manager, we have a list of restaurants accessible when the manager is logged in, and the view is different for the user and manager, and only the manager can see it. As can be seen from the figure, the manager is able to edit and delete a restaurant from the list. For editing, the user will be directed to the form for the restaurant details and data of the restaurants, such as the description of each restaurant, the manager's email, and the opening and closing hours.

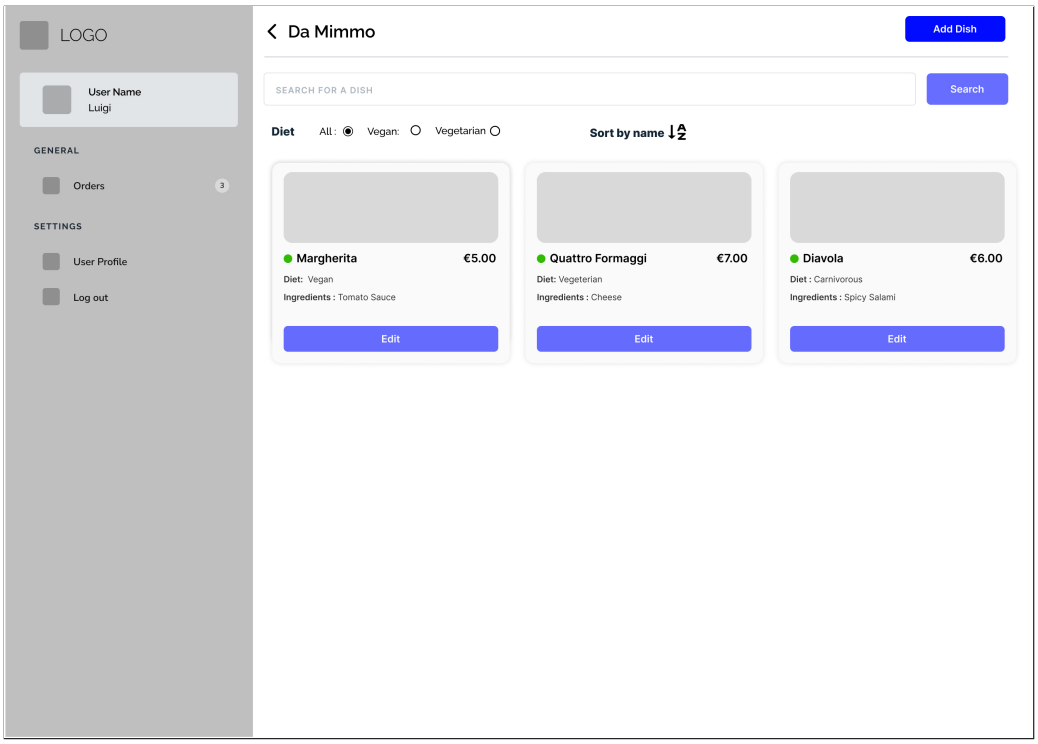


Figure 11: List of dishes from manager view.

The manager is able to view all the dishes and also make changes to each dish. This is included with the availability of adding and removing dishes. It is also feasible that both the administrator and manager have access to the dishes and can modify them. As you can see from Figure 11, there is also a feature for adding dishes. You will be directed to the form for adding the dish, and you will be able to edit the name, price, ingredients, and other attributes of the dish.

4.4 Customer View

The customer view is composed of the pages that are only visible to logged-in clients, therefore no guest can see these.

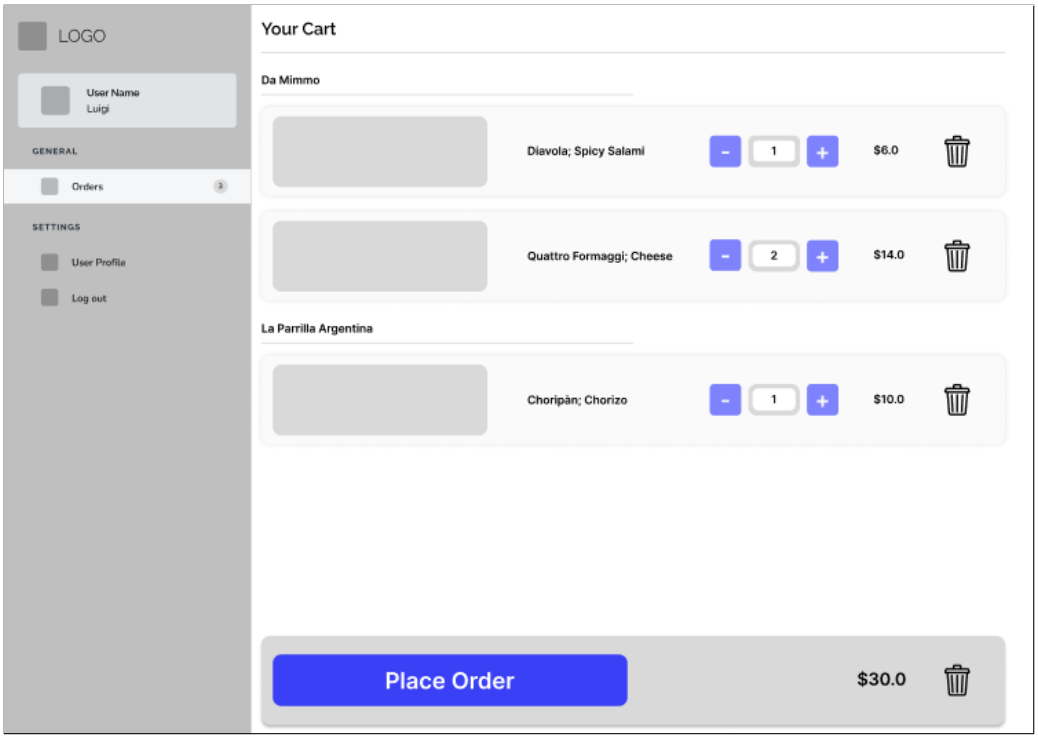


Figure 12: Shopping cart page that shows the list of dishes added to the order.

The virtual shopping cart page, figure 12, shows the current order for the user, listing all the selected dishes grouped by restaurant. The customer can add or subtract to the quantity of a dish, or can just delete it using the trash can button. The interface shows the price of each item and the total bill of the order. The user can complete the order by clicking on the "Place Order" button. Otherwise he can empty his cart by pressing the trash can icon near the bottom.

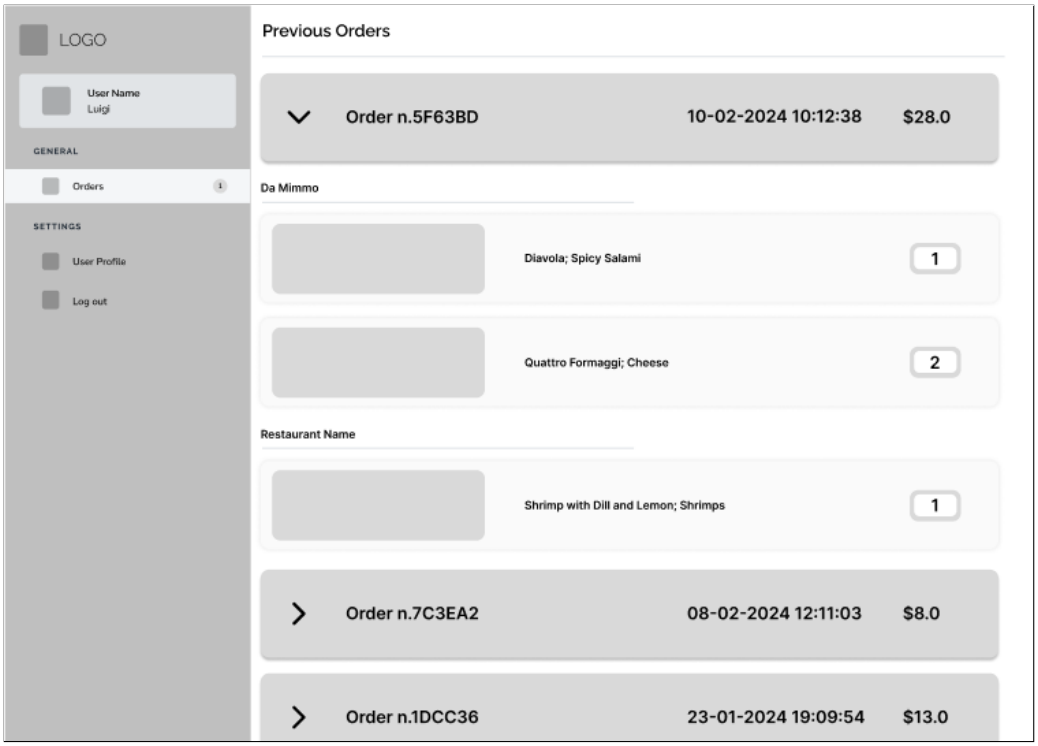


Figure 13: Previous orders' page.

The previous orders page, figure 13, shows the identifier, the date and the cost of each of the completed orders. By clicking on the drop down arrow on the left of an order, the customer can see the list of dishes. This page is designed to be scrollable, given that there could be many more completed orders than those that fit the screen.

5 Business Logic Layer

5.1 Class Diagram

We encountered a challenge with trying to fit the diagram of the project in a single figure, so we divided the original diagram into six smaller, more manageable schemes. Each of these schemes cover a different scope.

The first one, figure 14, shows the classes that we used to represent data in the business logic layer. The FullDish, FullOrder and DishIngredient classes are used to describe more complex data objects that do not directly match the entities stored in the database, as they are mainly used to contain data resulting from an SQL JOIN operation. The cuisine class does not extend the AbstractResource class because we didn't need to express it in JSON format.

All the following diagrams follow the basic REST paradigm that we implemented using a servlet dispatcher for the HTTP requests, some RR(REST Resource) classes for the main logic and DAO(Data Access Object) classes to communicate with the database. In particular, the second diagram, figure 15, contains all the classes relating to user account operations, like registering, logging-in and logging-out, updating user information and so on. Also, it shows the different authentication filter classes used on the requests. The third diagram, figure 16, shows the classes that perform operations relating to the restaurant entity, like creating a restaurant, deleting one or retrieving data about it. Then the fourth diagram, figure 17, shows the classes that perform operations relating to the dish entity, like creating a dish, deleting one or retrieving data about it. The fifth diagram, figure 18, shows the classes that are concerned with listing the cuisine types, which are useful for the advanced search tool. And lastly, the sixth diagram, in figure 19, shows the classes that perform operations relating to the food ordering process, like displaying the cart, adding dishes to the order, removing them or changing their quantity in the cart.

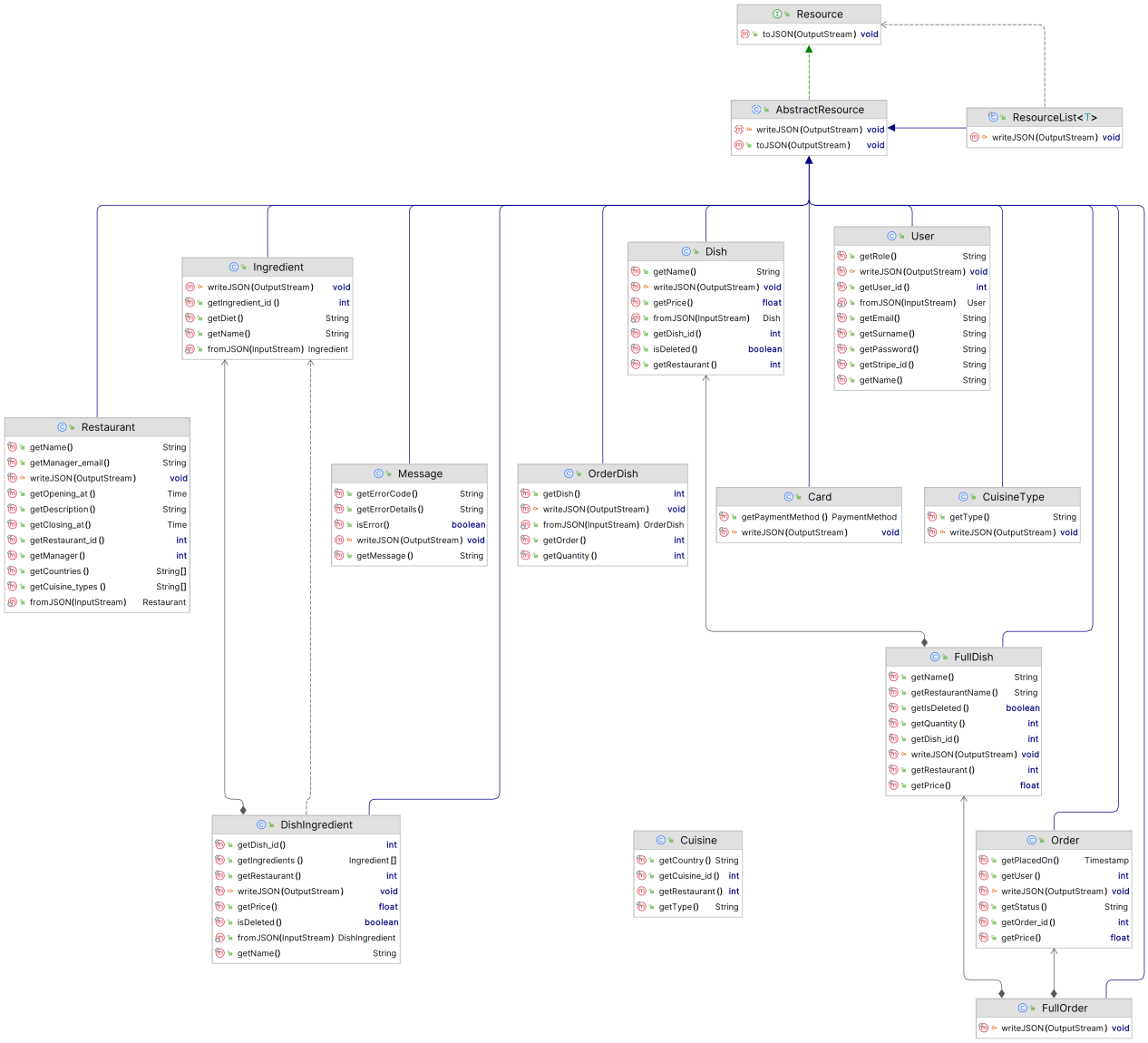


Figure 14: Class diagram of the database entities and their hierarchy.

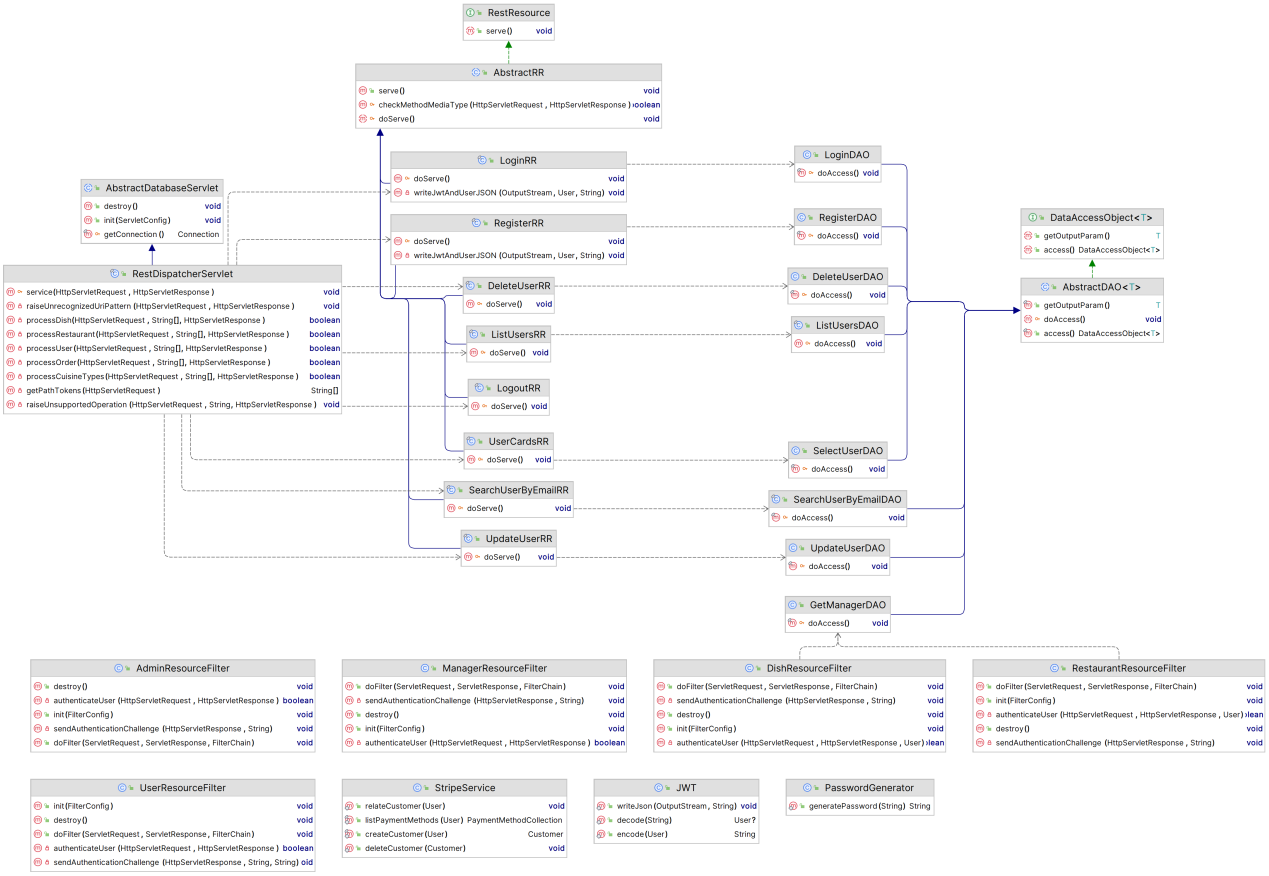


Figure 15: Class diagram of the user REST classes and the authentication filters.

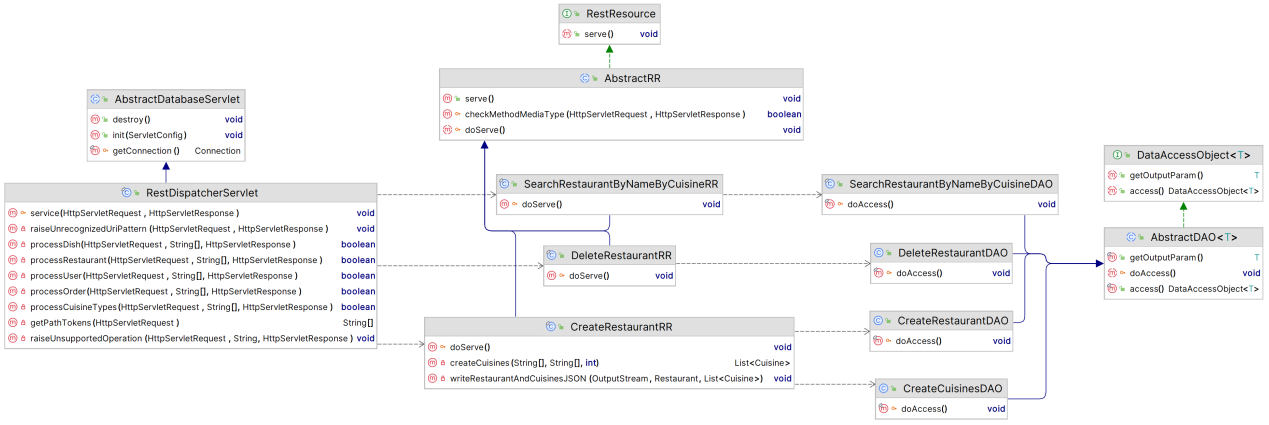


Figure 16: Class diagram of the restaurant REST classes.

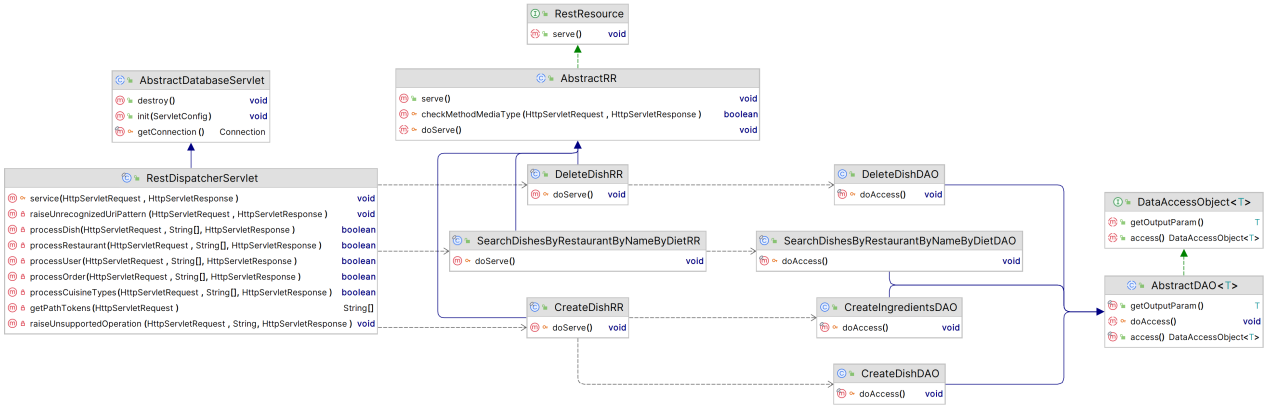


Figure 17: Class diagram of the dish REST classes.

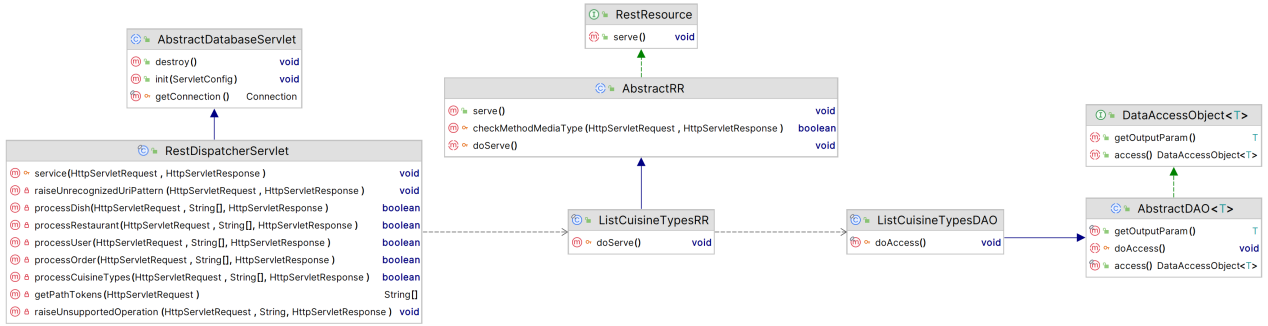


Figure 18: Class diagram of the cuisine REST classes.

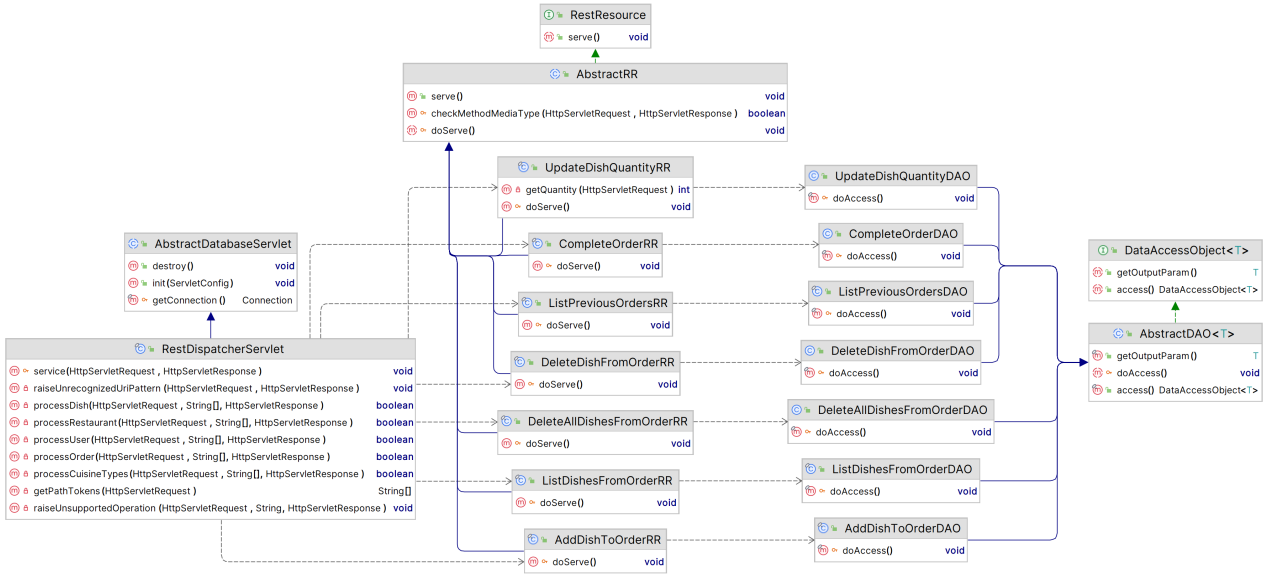


Figure 19: Class diagram of the order REST classes.

5.2 Sequence Diagram

The following two examples are representative of how resources are accessed in our application. The first one shows the login request and the second a generic request that needs authentication.

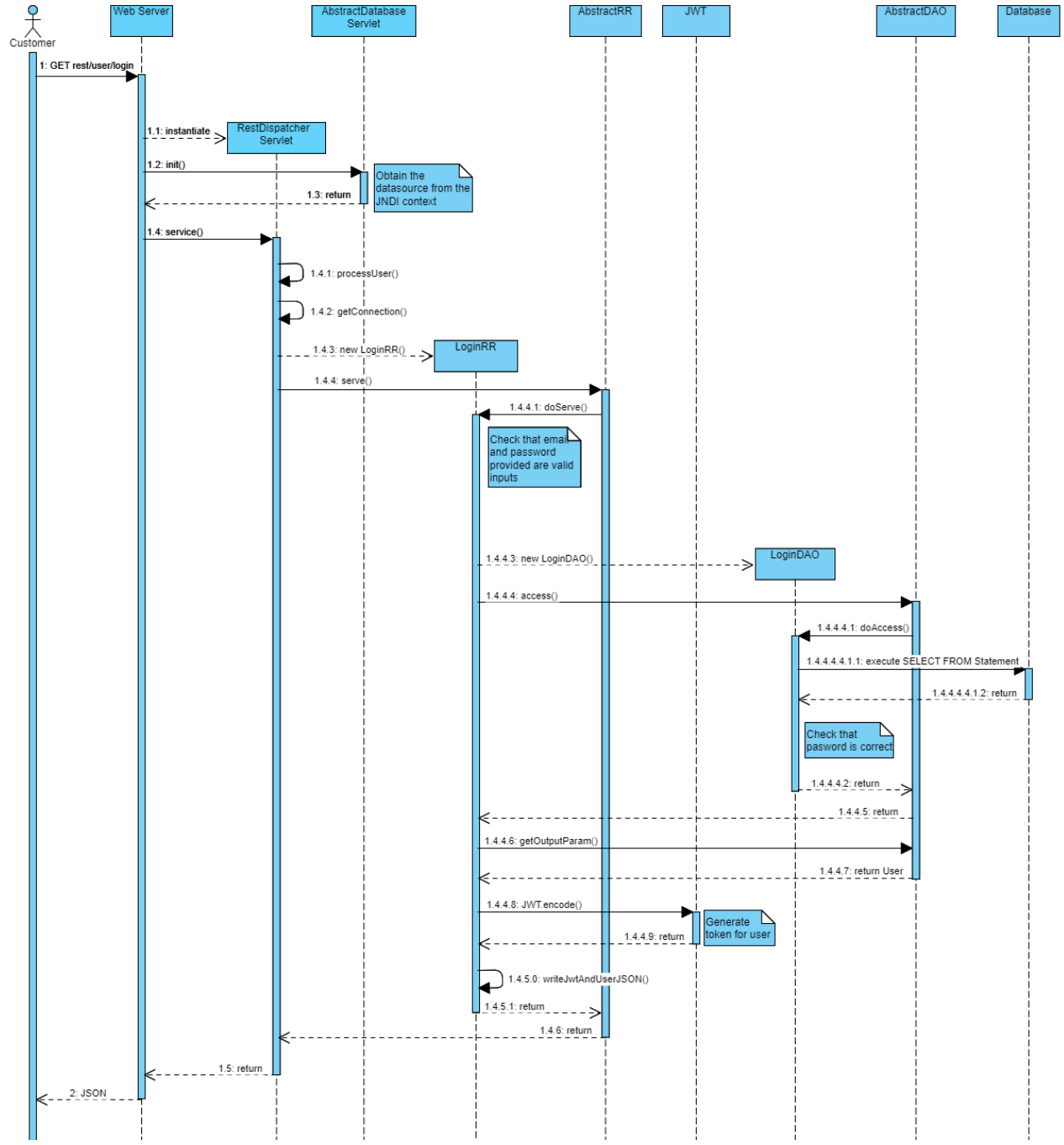


Figure 20: Sequence diagram for the GET */rest/user/login* request.

When a user sends a login request, by performing a GET HTTP request on the */rest/user/login* URI, the Tomcat web server checks if the URL is mapped to a filter. The login operation doesn't require authentication,

therefore the request is forwarded to the RestDispatcher servlet. The dispatcher is tasked with processing the URI and passing the request to the correct RR(REST Resource) class, which is LoginRR in this case. The RR validates the email and password as correctly formed and creates a new DAO(Data Access Object) to retrieve the user registration information from the database. After executing the SELECT FROM statement, the database returns the table row associated with the requested user(if it exists) and the DAO can now check that the password is correct by comparing it with the one in the database. At this point the control returns to LoginRR, that invokes the encode method from the JWT(JSON Web Token) class to extract the token for the user. The JWT and the corresponding User object are encoded in JSON format using the Jackson library and sent back as a response to the user.

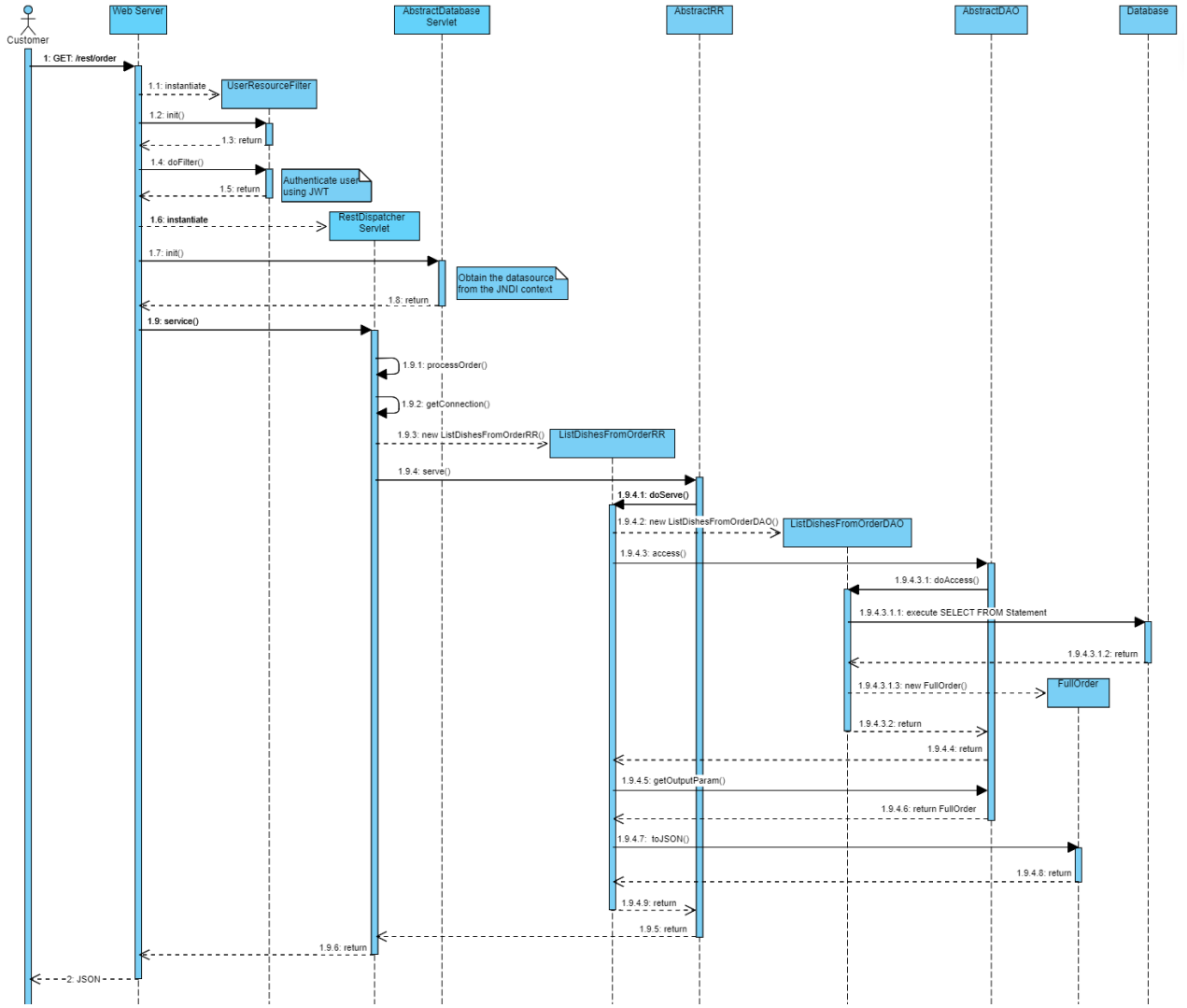


Figure 21: Sequence diagram for the GET */rest/order* request.

When a user asks to see the cart information, by performing a GET HTTP request on the *rest/order* URI, the Tomcat web server checks if the URL is mapped to a filter. In this case there is a mapping from *rest/order* to the UserResourceFilter class. The web server calls *init()* on the filter to set up and then *doFilter()* to authenticate the user. If the process goes as intented, the filter sets the attribute *auth_user* in the HTTP request to the

corresponding User object, otherwise it sends an authentication challenge to the user. At this point the user should be authenticated and the web server proceeds by instantiating the RestDispatcher servlet. The dispatcher is tasked with processing the URI and passing the request to the correct RR(REST Resource) class, which is ListDishesFromOrderRR in this case. The AbstractRR class, from which the ListDishesFromOrderRR class inherits, reads the *auth_user* attribute of the HTTP request into a protected field, when firstly instantiated. Therefore, the RR class can access it by inheritance and creates a new DAO to retrieve the list of dishes of the cart from the database. After executing the SELECT FROM statement, the database returns the table associated with the requested user's pending order(if it exists) and the DAO can now create a FullOrder object to contain the data of the Order, Order_Dish, Dish and Restaurant tables after an SQL JOIN operation. At this point the control returns to ListDishesFromOrderRR, that encodes the FullOrder object into a JSON format response. The message is then sent back to the user.

5.3 REST API Summary

All the rest resources are identified through the prefix */rest*. It will be omitted in the table below.

There are five kinds of filters:

1. AdminResourceFilter (*ARF*): Protects admin resources
2. UserResourceFilter (*URF*): Protects resources owned by the user
3. DishResourceFilter (*DRF*): Protects dishes from being modified by a manager other than the owner
4. ManagerResourceFilter (*MRF*): Protects manager operations
5. RestaurantResourceFilter (*RRF*): Protects restaurants from being modified by a manager other than the owner

The URIs can have optional parameters, we identify them by */parameterName/{?parameterValue}*. If a parameter has not to be passed, the whole block must be omitted.

URI	**Method**	**Description**	**Filter**
/restaurant/create	POST	Creates a restaurant	MRF
/restaurant/delete	DELETE	Deletes a restaurant	RRF
/restaurants/name/{?name}/cuisine-type/{?cuisine-type}	GET	Retrieves the restaurants information	None
/cuisine/types	GET	List all the cuisine types available in the platform	None
/order/dishes/{dishId}	POST	Adds {dishId} to the cart	URF
/order	PUT	Confirms the cart order	URF
/order/previous	GET	Lists the user previous orders	URF
/order	GET	Retrieves the user cart details	URF
/order/dishes/{dishId}	PUT	Updates the dish quantity in the cart	URF
/order/dishes/{dishId}	DELETE	Removes a dish from the cart	URF
/order/dishes	DELETE	Removes all dishes from the cart	URF
/dishes/restaurant_id/{?restaurant_id} /name/{?name}/diet/{?diet}	GET	Lists the dishes	None
/dish	POST	Creates a new dish	DRF

/dish	DELETE	Deletes a dish	DRF
/user/register	POST	Registers a customer user to the platform	None
/user/create	POST	Creates a manager user to the platform	ARF
/user	PUT	Updates user informations	URF
/user/login	GET	Login the user by retrieving a valid JWT token	None
/user/logout	GET	Logout the user by retrieving an invalid JWT token	URF
/user/delete	DELETE	Deletes a user from the platform	ARF
/user/list	GET	Lists the registered users in the platform	ARF
/user/email/{email}	GET	Retrieves the user informations	ARF
/user/cards	GET	Lists user credit cards	URF

Table 2: REST API description

5.4 REST Error Codes

Here the list of errors defined in the application.

Error Code	HTTP Status Code	Description
E5A1	INTERNAL_SERVER_ERROR	Unable to serve the REST request
ED00	INTERNAL_SERVER_ERROR	Unexpected error while accessing the database
EP00	BAD_REQUEST	Invalid input parameters for the requested resource
EP01	NOT_FOUND	Unknown resource requested
EP02	NOT_FOUND	Unexpected error while processing the REST resource
EP03	INTERNAL_SERVER_ERROR	Unrecognized pattern for the requested URI
EP04	METHOD_NOT_ALLOWED	Unsupported operation for the requested method type
ES00	INTERNAL_SERVER_ERROR	Cannot retrieve the user payment methods
EU00	INTERNAL_SERVER_ERROR	Unable to generate password encoding for the user
EU01	EMPTY_CART	The cart is empty, no information are available on it
EM00	BAD_REQUEST	Input media type not specified. Content-Type request header missing
EM01	BAD_REQUEST	Unsupported input media type. Resources are represented only in application/json
EF00	UNAUTHENTICATED	Cannot authenticate the user
EF01	UNAUTHORIZED	Unauthorized user for the requested resource

Table 3: REST API error codes

5.5 REST API Details

5.5.1 User

5.5.1.1 Register user

This endpoint will register a customer user to the platform.

- **URL:**

  /user/register

- **Method:**

  POST

- **URL Parameters:**

  None

- **Data Parameters:**

```
{
    "user":
    [
        {
            "email": "test10@gmail.com",
            "password": "7wZUY4+hQYK*bcyP*g",
            "name": "Daniel",
            "surname": "Carlesso",
            "role": "customer"
        }
    ]
}
```

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "jwt": {
        "access_token":
            "eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjksInJvbCI6ImN1c3
            RvbWVyIiwic3RyIjoiY3VzX1EwWEVLSDhDd2pNRXBKIiwiZGF0IjoxNzE0MzMzNDg1
            MTc0fQ.vNtpcCGGvfoikd6yvy4bORjgUq2Q9bqMnI-1F4duXLoNv _5itOah3WaEnf
            Rtst0leH0oBs7Y5GEQ75NpWMmngA",
        "token_type": "Bearer",
        "expires_in": 28800
    },
    "user": {
        "user_id": 9,
        "email": "test10@gmail.com",
        "name": "Daniel",
        "surname": "Carlesso",
        "stripe_id": "cus_Q0XEKH8CwjMEpJ",
        "role": "customer"
    }
}
```

- **Error Responses:**

  - **Code:** 500 Internal Server Error

    Content:

```
{
    "message": {
        "message": "Cannot create the user.",
        "error-code": "ED00",
        "error-details": "User already registered."
    }
}
```


- **Code:** 400 Bad Request

  Content:


```
{
    "message": {
        "message": "Cannot create the user: Invalid input parameters.",
        "error-code": "EP00",
        "error-details": "Invalid password."
    }
}
```


- **Sample Call:**


```
{
 "jwt": {
     "access_token":
         "eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjksInJvbCI6ImN1c3Rvb
         WVyIiwic3RyIjoiY3VzX1EwWEVLSDhDd2pNRXBKIiwiZGF0IjoxNzE0MzMzNDg1MTc0fQ
         .vNtpcCGGvfoikd6yvy4bORjgUq2Q9bqMnI-1F4duXLoNv_5itOah3WaEnfRtst0leH0o
         Bs7Y5GEQ75NpWMmngA",
     "token_type": "Bearer",
     "expires_in": 28800
 },
 "user": {
     "user_id": 9,
     "email": "test10@gmail.com",
     "name": "Daniel",
     "surname": "Carlesso",
     "stripe_id": "cus_Q0XEKH8CwjMEpJ",
     "role": "customer"
 }
}
```


5.5.1.2 Create user

This endpoint will create a manager user in the platform.

- **URL:**

  `/user/create`

- **Method:**

  `POST`

- **URL Parameters:**

  None

- **Data Parameters:**

```
{
    "user": {
        "email": "test4@gmail.com",
        "password": "7wZUY4+hQYK*bcyP*g",
        "name": "Luigi",
        "surname": "Frigione"
    }
}
```

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "jwt": {
        "access_token":
            "eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjQsInJvbCI6Im1hbmFnZXI
            iLCJzdHIiOiJjdXNfUTBISHllVHFHUTlBMFQiLCJkYXQiOjE3MTQyNzQxMjI3NDB9.tmQUn
            cw2vRxchlJowzwfp7rJSNeA8QvrZEXhYTF_e98Y3iggUqJKFZUr8IWHM7Ypa-txlQecFGN
            2xownNw",
        "token_type": "Bearer",
        "expires_in": 28800
    },
    "user": {
        "user_id": 4,
        "email": "test4@gmail.com",
        "name": "Luigi",
        "surname": "Frigione",
        "stripe_id": "cus_Q0HHyeTqGQ9A0T",
        "role": "manager"
    }
}
```

- **Error Responses:**

  - **Code:** 401 Unauthorized

    Content:

```
{
    "message": {
        "message": "Cannot authenticate the user.",
        "error-code": "EF00",
        "error-details": "Invalid JWT."
    }
}
```

  - **Code:** 401 Unauthorized

    Content:

```
{
    "message": {
        "message": "Unauthorized user for the requested resource.",
        "error-code": "EF01",
        "error-details": "EF01"
    }
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
    "message": {
        "message": "Cannot create the user.",
        "error-code": "ED00",
        "error-details": "User already registered."
    }
}
```

- **Code:** 400 Bad Request

  Content:

```
{
    "message": {
        "message": "Cannot create the user: Invalid input parameters.",
        "error-code": "EP00",
        "error-details": "Invalid password."
    }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/user/create",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjEsInJvbCI6ImFkbWluIiwic3Ry
        IjoiY3VzX1B3OHAyQlk2YkZYMm1iIiwiZGF0IjoxNzE0Mjc0MDk2MTkyfQ.2JrMhzazmE7qBH6A
        ASdIJj8C4pK0kjXcT9ZbGDRI5xfo6A_4IUA2oRzZwSzyrGXVEXB_ACCVV3GxONpDySlfvg"
  },
  "data": JSON.stringify({
    "user": {
      "email": "test4@gmail.com",
      "password": "7wZUY4+hQYK*bcyP*g",
      "name": "Luigi",
      "surname": "Frigione"
    }
  }),
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.1.3 Update user

This endpoint will update all the data about a user in the platform.

- **URL:**

  /user

- **Method:**

  PUT

- **URL Parameters:**

  None

- **Data Parameters:**

```
{
    "user":
    [
        {
            "user_id": 3,
            "email": "test15@gmail.com",
            "password": "7wZUY4+hQYK*bcyP*g",
            "name": "Daniel",
            "surname": "Carlesso",
            "role": "customer"
        }
    ]
}
```

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "user": {
        "user_id": 3,
        "email": "test15@gmail.com",
        "name": "Daniel",
        "surname": "Carlesso",
        "stripe_id": "cus_Pw8rJngyzWNrfy",
        "role": "customer"
    }
}
```

- **Error Responses:**

  - **Code:** 400 Bad Request

    Content:

```
{
    "message": {
        "message": "Cannot update the user. Invalid input parameters.",
        "error-code": "EP00",
        "error-details": "Invalid password."
    }
}
```

  - **Code:** 401 Unauthorized

    Content:

```
{
    "message": {
        "message": "Cannot authenticate the user.",
        "error-code": "EF01",
        "error-details": "JWT expired."
    }
}
```

- **Code:** 400 Bad Request

  Content:

```
{
    "message": {
        "message": "Cannot update the user",
        "error-code": "ED00",
        "error-details": "Cannot update user: unexpected DB error."
    }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/user",
  "method": "PUT",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN1c3Rv
        bWVyIiwic3RyIjoiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0Mjg0NjAyNzc
        4fQ._buJbO5joz8J8j-DlaoQfG7i7lHRrPae05n63TNPwMXfmSKF0zmyjS0exIQpILn
        VERLVAAD8fS4mCTBkmeYD7A"
  },
  "data": JSON.stringify({
    "user": [
      {
        "user_id": 3,
        "email": 2,
        "password": "7wZUY4+hQYK*bcyP*g",
        "name": "Daniel",
        "surname": "Carlesso",
        "role": "customer"
      }
    ]
  }),
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.1.4 User Login

This endpoint will login a user to the platform by retrieving a valid JWT token

- **URL:**

  /user/login

- **Method:**

  GET

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:


```
{
    "jwt": {
        "access_token":
            "eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjIsInJvbCI6Im1hbmFnZXI
            iLCJzdHIiOiJjdXNfUHc4cUNVWTg4V2gyMG4iLCJkYXQiOjE3MTQyODI5NDA2MDB9.FtoQG
            unPFNcmYupNfLS1gsdpPRpv3lb4YlbUB61AfTH8DvuI0ATXOsiXAHvi9P3EhpzoAtvmqBFF
            Lh88782Z0g",
        "token_type": "Bearer",
        "expires_in": 28800
    },
    "user": {
        "user_id": 2,
        "email": "manager.zero@studenti.unipd.it",
        "name": "Gino",
        "surname": "Bianchi",
        "stripe_id": "cus_Pw8qCUY88Wh20n",
        "role": "manager"
    }
}
```


- **Error Responses:**

  - **Code:** 400 Bad Request

    Content:


```
{
    "message": {
        "message": "Cannot log in the user: Invalid input parameters.",
        "error-code": "EP00",
        "error-details": "Invalid email address."
    }
}
```


  - **Code:** 400 Bad Request

    Content:


```
{
    "message": {
        "message": "Cannot log in the user: Invalid input parameters.",
        "error-code": "EP00",
```

```
            "error-details": "Invalid password."
        }
    }
```


- **Code:** 500 Internal Server Error

  Content:


```
    {
        "message": {
            "message": "Cannot create the user.",
            "error-code": "ED00",
            "error-details": "User not found."
        }
    }
```


- **Sample Call:**


```
var settings = {
  "url": "http://localhost/pl8s/rest/user/login",
  "method": "GET",
  "timeout": 0,
  "headers": {
    "Authorization": "Basic
        bWFuYWdlci56ZXJvQHN0dWRlbnRpLnVuaXBkLml0OkImc3RNYW5hZ2VyMjAyNA=="
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```


5.5.1.5 User Logout

This endpoint will logout the user by retrieving an invalid URF JWT token

- **URL:**

  /user/logout

- **Method:**

  GET

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "jwt": {
        "access_token": "invalidated",
        "token_type": "Bearer",
        "expires_in": 28800
    }
}
```

- **Error Response:**
  - **Code:** 500 Internal Server Error

    Content:

```
{
    "message": {
        "message": "Fatal error while logging out user.",
        "error-code": "ED00",
        "error-details": "Cannot log out user: Unexpected error."
    }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/user/logout",
  "method": "GET",
  "timeout": 0,
  "headers": {
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjIsInJvbCI6Im1hbmFnZXIiLCJzdHI
        iOiJjdXNfUHc4cUNVWTg4V2gyMG4iLCJkYXQiOjE3MTQyNjE1MTgyNjJ9.8q9hUBBFyssaR94A9xut
        jKG90LpP4mOGhvVjrzaFbPuSsU9sToBLZC-c1jvhxBrPXKdQyiojprTP_iNHu-ZFzQ"
  },
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.1.6 Delete user

This endpoint will delete a user from the platform. Only admins can access this API

- **URL:**

  /user/delete

- **Method:**

  DELETE

- **URL Parameters:**

  None

- **Data Parameters:**

```
{
    "user": {
        "user_id": 4
    }
}
```

- **Success Response:**
  - **Code:** 200 OK

    Content:

```
{
    "user": {
        "user_id": 4,
        "email": "test10@gmail.com",
        "name": "Luigi",
        "surname": "Frigione",
        "stripe_id": "cus_Q0UrcvJjv2CFTs",
        "role": "manager"
    }
}
```

- **Error Responses:**
  - **Code:** 401 Unauthorized

    Content:

```
{
    "message": {
        "message": "Cannot authenticate the user.",
        "error-code": "EF00",
        "error-details": "Bearer authentication is expected."
    }
}
```

  - **Code:** 500 Internal Server Error

    Content:

```
{
    "message": {
        "message": "Cannot delete the user.",
        "error-code": "ED00",
        "error-details": "User not found."
    }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/user/delete",
  "method": "DELETE",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjEsInJvbCI6ImFkbWluIiwic3RyI
        joiY3VzX1B3OHAyQlk2YkZYMm1iIiwiZGF0IjoxNzE0MzIwOTY0NzI5fQ.T1y3Lufb8wi1AxP6uy
        mJUrEDKo5dfgUe0seNIdsSw6yGQ_2R0awineJsMSa9TQu6lIy8dSH12JO9LPNzGtVJkg"
```

```
  },
  "data": JSON.stringify({
    "user": {
      "user_id": 4
    }
  }),
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.1.7 List user

This endpoint will list all the users registered in the platform. Only admins can access this API

- **URL:**

  /user/list

- **Method:**

  GET

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "users": [
        {
            "user": {
                "user_id": 2,
                "email": "manager.zero@studenti.unipd.it",
                "name": "Gino",
                "surname": "Bianchi",
                "stripe_id": "cus_Pw8qCUY88Wh20n",
                "role": "manager"
            }
        },
        {
            "user": {
                "user_id": 3,
                "email": "customer.zero@studenti.unipd.it",
                "name": "Mario",
                "surname": "Rossi",
```

```
                    "stripe_id": "cus_Pw8rJngyzWNrfy",
                    "role": "customer"
                }
            }
        ]
    }
```


- **Error Responses:**
  - **Code:** 500 Internal Server Error

    Content:


```
    {
        "message": {
            "message": "Cannot list user(s): unexpected database error.",
            "error-code": "ED00",
            "error-details": "Cannot list user(s): unexpected database error."
        }
    }
```


- **Sample Call:**


```
var settings = {
  "url": "http://localhost/pl8s/rest/user/list",
  "method": "GET",
  "timeout": 0,
  "headers": {
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjEsInJvbCI6ImFkbWluIiwic3Ry
        IjoiY3VzX1B3OHAyQlk2YkZYMm1iIiwiZGF0IjoxNzE0MzIwOTY0NzI5fQ.T1y3Lufb8wi1AxP6
        uymJ UrEDKo5dfgUe0seNIdsSw6yGQ_2R0awineJsMSa9TQu6lIy8dSH12JO9LPNzGtVJkg"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```


5.5.1.8 List user cards

This endpoint will list the user credit cards.

- **URL:**

  /user/cards

- **Method:**

  GET

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:


```
{
    "cards": [
        {
            "acss_debit": null,
            "affirm": null,
            "afterpay_clearpay": null,
            "alipay": null,
            "amazon_pay": null,
            "au_becs_debit": null,
            "bacs_debit": null,
            "bancontact": null,
            "billing_details": {
                "address": {
                    "city": null,
                    "country": null,
                    "line1": null,
                    "line2": null,
                    "postal_code": null,
                    "state": null
                },
                "email": null,
                "name": null,
                "phone": null
            },
            "blik": null,
            "boleto": null,
            "card": {
                "brand": "visa",
                "checks": {
                    "address_line1_check": null,
                    "address_postal_code_check": null,
                    "cvc_check": "pass"
                },
                "country": "GB",
                "description": null,
                "display_brand": "visa",
                "exp_month": 12,
                "exp_year": 2025,
                "fingerprint": "40woMAkf0kx6deDl",
                "funding": "credit",
                "iin": null,
                "issuer": null,
                "last4": "0000",
                "networks": {
                    "available": [
                        "visa"
                    ],
                    "preferred": null
                },
```

```
                    "three_d_secure_usage": {
                        "supported": true
                    },
                    "wallet": null
                },
                "card_present": null,
                "cashapp": null,
                "created": 1713292426,
                "customer": "cus_Pw8rJngyzWNrfy",
                "customer_balance": null,
                "eps": null,
                "fpx": null,
                "giropay": null,
                "grabpay": null,
                "id": "pm_1P6GpCIlIIouy5CbqMLLY0go",
                "ideal": null,
                "interac_present": null,
                "klarna": null,
                "konbini": null,
                "link": null,
                "livemode": false,
                "metadata": {},
                "mobilepay": null,
                "object": "payment_method",
                "oxxo": null,
                "p24": null,
                "paynow": null,
                "paypal": null,
                "pix": null,
                "promptpay": null,
                "radar_options": {
                    "session": null
                },
                "revolut_pay": null,
                "sepa_debit": null,
                "sofort": null,
                "swish": null,
                "type": "card",
                "us_bank_account": null,
                "wechat_pay": null,
                "zip": null
            }
        ]
    }
```


- **Error Responses:**
  - **Code:** 401 Unauthorized

    Content:


```
    {
        "message": {
            "message": "Cannot authenticate the user.",
            "error-code": "EF00",
            "error-details": "No authorization header sent."
        }
    }
```


  - **Code:** 401 Unauthorized

Content:

```
{
      "message": {
          "message": "Cannot authenticate the user.",
          "error-code": "EF00",
          "error-details": "Invalid JWT."
      }
  }
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "message": {
          "message": "Fatal error while retrieving user.",
          "error-code": "ED00",
          "error-details": "Cannot retrieve the user: Unexpected db error."
      }
  }
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "message": {
          "message": "Cannot retrieve the user payment methods.",
          "error-code": "ES00",
          "error-details": "Cannot retrieve the user payment methods."
      }
  }
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/user/cards",
  "method": "GET",
  "timeout": 0,
  "headers": {
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN1c3RvbWVyIi
        wic3R yIjoiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0MzI5NTYzODk1fQ.dy9GSq4
        X3sGora MkCnvD4bZdtlPtiMjBpPxdN6Ul2I5tgreCK_pRn6K0UXe2AXIp3XMQ4p-W9AJ8W79
        vtFdLng"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.1.9 Search users

This endpoint will search a user by his email address and retrieve all his information. Partial addresses are allowed: e.g "john" instead of "john.dough@email.com". Only admins can access this API

- **URL:**

  /user/email/{email}

- **Method:**

  GET

- **URL Parameters:**

  Required:

  email=[string]

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "users": [
        {
            "user": {
                "user_id": 2,
                "email": "manager.zero@studenti.unipd.it",
                "name": "Gino",
                "surname": "Bianchi",
                "stripe_id": "cus_Pw8qCUY88Wh20n",
                "role": "manager"
            }
        }
    ]
}
```

- **Error Response:**

  - **Code:** 400 Bad Request

    Content:

```
{
  "message": {
    "message": "Cannot search for user(s): unexpected database error.",
    "error-code": "ED00",
    "error-details": "Cannot search for user(s): unexpected database error."
  }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/user/email/manager",
  "method": "GET",
```

```
  "timeout": 0,
  "headers": {
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjEsInJvbCI6ImFkbWluIiwic3RyIjoi
        Y3VzX1B3OHAyQ1k2YkZYMm1iIiwiZGF0IjoxNzE0MzIwOTY0NzI5fQ.T1y3Lufb8wi1AxP6uymJUrED
        Ko5dfgUe0seNIdsSw6yGQ_2R0awineJsMSa9TQu6lIy8dSH12JO9LPNzGtVJkg"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.2 Cuisine

5.5.2.1 Cuisine types

This endpoint retrieve the list of cuisine types available in the platform.

- **URL:**

  `/cuisine/types`

- **Method:**

  `GET`

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "cuisinetypes": [
        {
            "CuisineType": {
                "type": "insalata"
            }
        },
        {
            "CuisineType": {
                "type": "steakhouse"
            }
        }
    ]
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/cuisine/types",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.3 Dish

5.5.3.1 Create Dish

This endpoint will create a new dish for a restaurant.

- **URL:**

  `/dish`

- **Method:**

  `POST`

- **URL Parameters:**

  None

- **Data Parameters:**

```
{
    "dish": {
        "name": "pasticcio",
        "price": 15,
        "restaurant": 2,
        "ingredients":
            [
                {"name": "uova", "diet": "vegetarian"},
                {"name": "besciamella", "diet": "vegetarian"},
                {"name": "ragu", "diet": "carnivorous"},
                {"name": "radicchio", "diet": "vegan"}
            ]
    }
}
```

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
    "dish": {
        "dish_id": 17,
        "name": "pasticcio",
```

```
            "price": 15.0,
            "isDeleted": false,
            "restaurant": 2,
            "ingredients": [
                {
                    "ingredient_id": 38,
                    "name": "uova",
                    "diet": "vegetarian"
                },
                {
                    "ingredient_id": 39,
                    "name": "besciamella",
                    "diet": "vegetarian"
                },
                {
                    "ingredient_id": 40,
                    "name": "ragu",
                    "diet": "carnivorous"
                },
                {
                    "ingredient_id": 41,
                    "name": "radicchio",
                    "diet": "vegan"
                }
            ]
        }
    }
```


- **Error Response:**
  - **Code:** 401 Unauthorized

    Content:


```
     "message": {
      "message": "Unable to retrieve the manager of the restaurant that serves this
          dish.",
      "error-code": "EP00",
      "error-details": "Restaurant not found."
    }
```


    OR
  - **Code:** 500 Internal Server Error

    Content:


```
    "message": {
      "message": "Cannot create the dish.",
      "error-code": "ED00",
      "error-details": "Cannot create dish.."
    }
```


- **Sample Call:**


```
var settings = {
  "url": "http://localhost/pl8s/rest/dish",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
```

```
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjIsInJvbCI6Im1hbmFnZXIiLCJz
        dHIiOiJjdXNfUHc4cUNVWTg4V2gyMG4iLCJkYXQiOjE3MTQyODE5NjEwNzV9.aQ-zURTFf9g7D8
        5NxijemKgeKmdnfyTPFL2XmpqwoDJCgirZv1P_7sZ4SsXcZ8UjN_ONSSyYwG98SvyxvorNdg"
  },
  "data": JSON.stringify({
    "dish": {
      "name": "pasticcio",
      "price": 15,
      "restaurant": 2,
      "ingredients": [
        {
          "name": "uova",
          "diet": "vegetarian"
        },
        {
          "name": "besciamella",
          "diet": "vegetarian"
        },
        {
          "name": "ragu",
          "diet": "carnivorous"
        },
        {
          "name": "radicchio",
          "diet": "vegan"
        }
      ]
    }
  }),
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
});
```

5.5.3.2 Delete dish

This endpoint will soft delete a dish from a restaurant.

- **URL:**

  /dish

- **Method:**

  DELETE

- **URL Parameters:**

  None

- **Data Parameters:**

  ```
  {
  ```

```
  "dish": {
    "dish_id": 15,
    "restaurant": 2
  }
}
```

- **Success Response:**
  - **Code:** 200 OK

    Content:

```
{
  "dish": {
    "dish_id": 15,
    "name": "pasticcio",
    "price": 15.0,
    "isDeleted": true,
    "restaurant": 2
  }
}
```

- **Error Responses:**
  - **Code:** 500 Internal Server Error

    Content:

```
{
  "message": {
    "message": "Cannot delete the dish.",
    "error-code": "ED00",
    "error-details": "Dish not found."
  }
}
```

  - **Code:** 400 Bad Request

    Content:

```
    "message": {
      "message": "Cannot delete the dish: Invalid input parameters.",
      "error-code": "EP00",
      "error-details": "Cannot delete the dish: Invalid input parameters.."
    }
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/dish",
  "method": "DELETE",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjIsInJvbCI6Im1hbmFnZXIiLCzd
        HIiOiJjdXNfUHc4cUNVWTg4V2gyMG4iLCJkYXQiOjE3MTQyNjE3MDA1MjV9.-shbruJiCHs-8md
        vaswMNs-9CmzyKGNSkX0vCD8IuJnE8jLgfTjGAT-LdrAHEd5uKS8Hpp0Wn-VQ-OSbyqrEGQ"
  },
  "data": JSON.stringify({
```

```
    "dish": {
      "dish\textunderscore id": 16,
      "restaurant": 2
    }
  }),
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.3.3 Search dishes

This endpoint will search for dishes according to the different parameters provided.

- **URL:**

  /dishes

- **Method:**

  GET

- **URL Parameters:**

  Optional:

  restaurant_id=[integer]

  diet=[string]

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
"dishes": [
    {
        "dish": {
            "dish_id": 4,
            "name": "Asado",
            "price": 15.0,
            "isDeleted": false,
            "restaurant": 3
        }
    },
    {
        "dish": {
            "dish_id": 6,
```

```
                "name": "Choripan",
                "price": 10.0,
                "isDeleted": false,
                "restaurant": 3
            }
        }
    }
```


- **Error Response:**
  - **Code:** 400 Bad Request

    Content:


```
{
  "message": {
    "message": "Cannot read the restaurant: wrong format for URI
        /restaurant/{restaurant_id}.",
    "error-code": "EP00",
    "error-details": "Cannot read the restaurant: wrong format for URI
        /restaurant/{restaurant_id}."
  }
}
```


    OR

  - **Code:** 500 Internal Server Error

    Content:


```
{
  "message": {
    "message": "Cannot list dish(es): unexpected database error.",
    "error-code": "ED00",
    "error-details": "Cannot list dish(es): unexpected database error."
  }
}
```


- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/dishes/",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.4 Order

5.5.4.1 Add dish to the cart

This endpoint will add the selected dish to the cart.

- **URL:**

  `/order/dishes/{dishId}`

- **Method:**

  POST

- **URL Parameters:**

  Required:

  dishId=[integer]

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
  "order_dish": {
    "order": 9,
    "dish": 6,
    "quantity": 1
  }
}
```

- **Error Responses:**

  - **Code:** 401 Unauthorized

    Content:

```
{
  "message": {
    "message": "Cannot authenticate the user.",
    "error-code": "EF00",
    "error-details": "No authorization header sent."
  }
}
```

  OR

  - **Code:** 500 Internal Server Error

    Content:

```
{
  "message": {
    "message": "Cannot create the OrderDish.",
    "error-code": "ED00",
    "error-details": "OrderDish relation already registered."
  }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/order/dishes/6",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN1c3
        RvbWVyIiwic3RyIjoiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0MzQyNjA
        xNjMwfQ.RrH8ojpudntUVEZVnbvVw1GpUEU56tdnhnCSrnCqoHgUAwMMxXJDcTMfK
        azQmrIZ_OSfges7-lECny6p-t5ITA"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.4.2 Update cart dish quantity

This endpoint will update the dish quantity.

- **URL:**

  /order/dishes/{dishId}

- **Method:**

  PUT

- **URL Parameters:**

  Required:

  dishId=[integer]

- **Data Parameters:**

```
{
    "quantity": 4
}
```

- **Success Response:**
  - **Code:** 200 OK

    Content:

```
{
  "order_dish": {
    "order": 9,
    "dish": 6,
    "quantity": 4
  }
}
```

- **Error Responses:**
  - **Code:** 401 Unauthorized

    Content:

```
{
  "message": {
    "message": "Cannot authenticate the user.",
    "error-code": "EF00",
    "error-details": "No authorization header sent."
  }
}
```

OR

  - **Code:** 400 Bad Request

    Content:

```
{
  "message": {
    "message": "Cannot update the OrderDish. Invalid input parameters.",
    "error-code": "EP00",
    "error-details": "Quantity must be greater than zero."
  }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/order/dishes/6",
  "method": "PUT",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN1c3RvbWVyI
        iwic3RyIjoiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0MzQyNjAxNjMwfQ.RrH8oj
        pudntUVEZVnbvVw1GpUEU56tdnhnCSrnCqoHgUAwMMxXJDcTMfKazQmrIZ_OSfges7-lECny
        6p-t5ITA"
  },
  "data": JSON.stringify({
    "quantity": 4
  }),
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.4.3 Delete dish from the cart

This endpoint will delete the selected dish from the cart.

- **URL:**

  `/order/dishes/{dishId}`

- **Method:**

  DELETE

- **URL Parameters:**

  Required:

  dishId=[integer]

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
  "order_dish": {
    "order": 11,
    "dish": 7,
    "quantity": 1
  }
}
```

- **Error Responses:**

  - **Code:** 401 Unauthorized

    Content:

```
{
  "message": {
    "message": "Cannot authenticate the user.",
    "error-code": "EF00",
    "error-details": "No authorization header sent."
  }
}
```

  OR

  - **Code:** 500 Internal Server Error

    Content:

```
{
  "message": {
    "message": "Cannot delete the OrderDish.",
    "error-code": "ED00",
    "error-details": "OrderDish not found."
  }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/order/dishes/7",
  "method": "DELETE",
  "timeout": 0,
  "headers": {
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN1c3Rvb
        WVyIiwic3RyIjoiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0MzQyNjAxNjMwf
        Q.RrH8ojpudntUVEZVnbvVw1GpUEU56tdnhnCSrnCqoHgUAwMMxXJDcTMfKazQmrIZ_O
        Sfges7-lECny6p-t5ITA"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.4.4 Empty the cart

This endpoint will empty the cart.

- **URL:**

  /order/dishes

- **Method:**

  DELETE

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

    ```
    {
      "order": {
        "order_id": 11,
        "price": 0.0,
        "placedOn": "2024-04-28 14:49:51.397295",
        "status": "pending",
        "user": 3
      }
    }
    ```

- **Error Responses:**

  - **Code:** 401 Unauthorized

Content:

```
{
  "message": {
    "message": "Cannot authenticate the user.",
    "error-code": "EF00",
    "error-details": "No authorization header sent."
  }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/order/dishes",
  "method": "DELETE",
  "timeout": 0,
  "headers": {
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN1cRvbW
        VyIiwic3RyIjoiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0MzQyN jAxNjMwf
        Q.RrH8ojpudntUVEZVnbvVw1GpUEU56tdnhnCSrnCqoHgUAwMMxXJDcT MfKazQmrIZ_
        OSfges7-lECny6p-t5ITA"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.4.5 Complete order

This endpoint will confirm the cart order.

- **URL:**

  /order

- **Method:**

  PUT

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
  "order": {
    "order_id": 9,
    "price": 40.0,
    "placedOn": "2024-04-28 14:49:51.397295",
    "status": "completed",
    "user": 3
  }
}
```

- **Error Responses:**
  - **Code:** 401 Unauthorized

    Content:

```
{
  "message": {
    "message": "Cannot authenticate the user.",
    "error-code": "EF00",
    "error-details": "No authorization header sent."
  }
}
```

OR

  - **Code:** 500 Internal Server Error

    Content:

```
{
  "message": {
    "message": "Cannot complete the Order.",
    "error-code": "ED00",
    "error-details": "Order not found."
  }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/order",
  "method": "PUT",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/json",
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN1c3RvbWVyIiwic3RyI
        joiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0M zQyNjAxNjMwfQ.RrH8ojpudntUVEZVnbvVw
        1GpUEU56tdnhnCSrnCqoHgUAwMMxXJDcTMfKazQm rIZ_OSfges7-lECny6p-t5ITA"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.4.6 Previous orders

This endpoint will list all the previous orders.

- **URL:**

  `/order/previous`

- **Method:**

  `GET`

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:


```
{
  "previous orders": [
    {
      "order": {
        "order_id": 1,
        "price": 22.0,
        "placedOn": "2024-04-25 06:35:47.873674",
        "status": "completed",
        "user": 3,
        "dishes": [
          {
            "dish": {
              "dish_id": 2,
              "name": "Diavola",
              "price": 6.0,
              "restaurant": 2,
              "quantity": 1
            }
          },
          {
            "dish": {
              "dish_id": 7,
              "name": "Shrimp with Dill and Lemon",
              "price": 8.0,
              "restaurant": 4,
              "quantity": 2
            }
          }
        ]
      }
    },
```

```
{
  "order": {
    "order_id": 2,
    "price": 28.0,
    "placedOn": "2024-04-25 06:42:51.528244",
    "status": "completed",
    "user": 3,
    "dishes": [
      {
        "dish": {
          "dish_id": 1,
          "name": "Margherita",
          "price": 5.0,
          "restaurant": 2,
          "quantity": 1
        }
      },
      {
        "dish": {
          "dish_id": 5,
          "name": "Empanadas",
          "price": 8.0,
          "restaurant": 3,
          "quantity": 2
        }
      },
      {
        "dish": {
          "dish_id": 9,
          "name": "Potato, Garlic aioli and Crispy Shallots",
          "price": 7.0,
          "restaurant": 4,
          "quantity": 1
        }
      }
    ]
  }
 }
]
}
```


- **Error Responses:**
  - **Code:** 401 Unauthorized

    Content:


```
{
  "message": {
    "message": "Cannot authenticate the user.",
    "error-code": "EF00",
    "error-details": "No authorization header sent."
  }
}
```


- **Sample Call:**


```
var settings = {
  "url": "http://localhost/pl8s/rest/order/previous",
  "method": "GET",
```

```
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.4.7 List dishes in the cart

This endpoint will list all the dishes in the cart.

- **URL:**

  /order

- **Method:**

  GET

- **URL Parameters:**

  None

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK
    Content:

```
{
  "order": {
    "order_id": 11,
    "price": 10.0,
    "placedOn": "2024-04-28 14:49:51.397295",
    "status": "pending",
    "user": 3,
    "dishes": [
      {
        "dish": {
          "dish_id": 6,
          "name": "Choripan",
          "price": 10.0,
          "restaurant": 3,
          "quantity": 1
        }
      }
    ]
  }
}
```

  OR

- **Code:** 200 OK

  Content:

```
{
  "message": {
    "message": "Empty cart.",
    "error-code": "EU01"
  }
}
```

- **Error Responses:**

  - **Code:** 401 Unauthorized

    Content:

```
{
  "message": {
    "message": "Cannot authenticate the user.",
    "error-code": "EF00",
    "error-details": "No authorization header sent."
  }
}
```

- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/order",
  "method": "GET",
  "timeout": 0,
  "headers": {
    "Authorization": "Bearer
        eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjMsInJvbCI6ImN
        1c3RvbWVyIiwic3RyIjoiY3VzX1B3OHJKbmd5eldOcmZ5IiwiZGF0IjoxNzE0M
        zQyNjAxNjMwfQ.RrH8ojpudntUVEZVnbvVw1GpUEU56tdnhnCSrnCqoHgUAwMM
        xXJDcTMfKazQmrIZ_OSfges7-lECny6p-t5ITA"
  },
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

5.5.5 Restaurant

5.5.5.1 Create restaurant

This endpoint will create a new restaurant in the platform.

- **URL:**

  `/restaurant/create`

- **Method:**

  `POST`

- **URL Parameters:**

  None

- **Data Parameters:**

```
{
    "restaurant": {
        "name": "Da Pilio e Frapiero",
        "description": "Buonissimo a Montegrotto",
        "opening_at": "17:00:00",
        "closing_at": "23:00:00",
        "countries": [
            "Italy",
            "Malta"
        ],
        "cuisine_types": [
            "Panini",
            "pasta",
            "salame",
            "insalata"
        ]
    }
}
```

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
  "restaurant": {
    "name": "Da Pilio e Frapiero",
    "description": "Buonissimo a Montegrotto",
    "opening_at": "17:00:00",
    "closing_at": "23:00:00",
    "countries": [
      "Italy",
      "Malta"
    ],
    "cuisine_types": [
      "Panini",
      "pasta",
      "salame",
      "insalata"
    ]
  }
}
```

- **Error Response:**

  - **Code:** 400 Bad Request

    Content:

```
{
  "message": {
    "message": "Input media type not specified.",
    "error-code": "EM00",
```

```
        "error-details": "Content-Type request header missing."
      }
    }
```


OR

- **Code:** 500 Internal Server Error

  Content:


```
    {
      "message": {
        "message": "Unable to serve the REST request: CREATE_RESTAURANT.",
        "error-code": "E5A1",
        "error-details": "Unable to parse JSON: no Restaurant object found."
      }
    }
```


- **Sample Call:**


```
var settings = {
    "url": "http://localhost/pl8s/rest/restaurant/create",
    "method": "POST",
    "timeout": 0,
    "headers": {
        "Content-Type": "application/json",
        "Authorization": "Bearer
            eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjIsInJvbCI6Im1hbmFnZXIiLC
            JzdHIiOiJjdXNfUHc4cUNVWTg4V2gyMG4iLCJkYXQiOjE3MTQyMjk0NzEzNTh9.83I0YU5yV5
            LJA4m4O_VQ1oPtUao40x76nV5FUzocGCOde0ijfe0cVq7BkSvre9--KsgNqgg2eX6jWpquIDB
            MAA"
    },
    "data": JSON.stringify({
        "restaurant": {
            "name": "Da Pilio e Frapiero",
            "description": "Buonissimo a Montegrotto",
            "opening_at": "17:00:00",
            "closing_at": "23:00:00",
            "countries": [
                "Italy",
                "Malta"
            ],
            "cuisine_types": [
                "Panini",
                "pasta",
                "salame",
                "insalata"
            ]
        }
    }),
};

$.ajax(settings).done(function (response) {
    console.log(response);
});
```


5.5.5.2 Delete restaurant

This endpoint will delete an existing restaurant.

- **URL:**

  `/restaurant/delete`

- **Method:**

  `DELETE`

- **URL Parameters:**

  None

- **Data Parameters:**

```
{
  "restaurant": {
    "restaurant_id": 15
  }
}
```

- **Success Response:**

  - **Code:** 200 OK

    Content:

```
{
  "restaurant": {
    "restaurant_id": 15,
    "name": "Da Pilio e Frapiero",
    "description": "Buonissimo a Montegrotto",
    "manager": 2,
    "opening_at": "17:00:00",
    "closing_at": "23:00:00",
    "countries": [],
    "cuisine_types": []
  }
}
```

- **Error Response:**

  - **Code:** 400 Bad Request

    Content:

```
{
  "message": {
    "message": "Unable to retrieve the manager of the restaurant.",
    "error-code": "EP00",
    "error-details": "Restaurant not found."
  }
}
```

- **Sample Call:**

```
var settings = {
    "url": "http://localhost/pl8s/rest/restaurant/delete",
    "method": "DELETE",
    "timeout": 0,
    "headers": {
        "Content-Type": "application/json",
        "Authorization": "Bearer
            eyJhcHAiOiJQTDhTIiwiYWxnIjoiSFM1MTIifQ.eyJ1aWQiOjIsInJvbCI6Im1hbmFnZXIiLCJ
            zdHIiOiJjdXNfUHc4cUNVWTg4V2gyMG4iLCJkYXQiOjE3MTQyNTgzOTA0MzV9.VB8_-34BvEUc
            sM6gVZl0C7q8W3SX767BErL9Nx_1rKluRrwkxi8Oy_9loyRLah6faQaO2hsQa3uxNBDpZYUjBQ"
    },
    "data": JSON.stringify({
        "restaurant": {
            "restaurant_id": 15
        }
    }),
};

$.ajax(settings).done(function (response) {
    console.log(response);
});
```

5.5.5.3 Search restaurants

This endpoint will search for registered restaurants.

- **URL:**

  /restaurants

- **Method:**

  GET

- **URL Parameters:**

  Optional:

  name=[string]

  cuisine_type=[string]

- **Data Parameters:**

  None

- **Success Response:**

  - **Code:** 200 OK

    Content:

    ```
    {
      "restaurants": [
        {
    ```

```
      "restaurant": {
        "restaurant_id": 2,
        "name": "Da Mimmo",
        "description": "Where the authentic flavors of Napoli come to life in every
            slice.",
        "manager": 2,
        "opening_at": "16:00:00",
        "closing_at": "22:00:00",
        "countries": [
          "Italy"
        ],
        "cuisine_types": [
          "pizza"
        ]
      }
    },
    {
      "restaurant": {
        "restaurant_id": 13,
        "name": "Da Pilio e Frapiero",
        "description": "Buonissimo a Montegrotto",
        "manager": 2,
        "opening_at": "17:00:00",
        "closing_at": "23:00:00",
        "countries": [
          "Italy",
          "Malta",
          null
        ],
        "cuisine_types": [
          "Panini",
          "insalata",
          "pasta",
          "salame"
        ]
      }
    }
  ]
}
```


- **Sample Call:**

```
var settings = {
  "url": "http://localhost/pl8s/rest/restaurants",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

6 Group Members Contribution

Manuel Antonutti contributed to this project by:

1. Helping in the ER schema development for the database.
2. Creating the mock up for the Cart and PreviousOrders pages in the customer view.
3. Creating the FullDish and FullOrder classes in the *dbentities* folder.
4. Implementing the ListDishesFromOrder RR and DAO classes.
5. Implementing the ListPreviousOrders RR and DAO classes.
6. Implementing the AddDishToOrder RR and DAO classes.
7. Implementing the UpdateDishQuantity RR and DAO classes.
8. Implementing the DeleteDishFromOrder RR and DAO classes.
9. Implementing the DeleteAllDishesFromOrder RR and DAO classes.
10. Implementing the CompleteOrder RR and DAO classes.
11. Implementing all the SQL procedures and functions in *festival-query.sql* for the DAOs above.
12. Providing the class diagrams for the project.
13. Providing the sequence diagrams for the project.

Daniel Carlesso contributed to this project by :

1. Helping in the ER schema development and doing the implementation.
2. Implementing the initialization of the db and the queries (not all).
3. Helping with the mocking in general and focusing more on the user functionalities
4. Implementing the Login/Register/Logout RR and DAO + PasswordGenerator class
5. Implementing the UpdateUser/DeleteUser RR and DAO
6. Implementing the CreateRestaurant/DeleteRestaurant RR and DAO
7. Implementing the CreateDish/DeleteDish RR and DAO
8. Implementing the resource filters + JWT class
9. Implementing the following DAOs used by other RR: CreateCuisinesDAO, CreateIngredientsDAO, GetManagerDAO
10. Filling DLL in the report
11. Filling the admin.tex file for the presentation logic layer

Luigi Frigione contributed to this project by:

1. Helping in the ER schema development for the database.
2. Designing and implementing the docker configuration
3. Designing and implementing the application database connection with a special user with limited permissions
4. Implementing the first version of ListRestaurants API
5. Studying and configuring Stripe integration for online payments
6. Implementing list user credit cards API

7. Designing payment API but not implementing it yet since stripe JavaScript library is needed.
8. Modifying user register API to be integrated with stripe (on registration we create an associate customer entity on the Stripe ecosystem)
9. Debugging and solving all the Javadoc errors
10. Writing the README.md file of the report with the instructions to build and navigate the application
11. Testing the application
12. Writing the documentation of the REST API summary in the HW1 report
13. Writing the documentation of the REST API details for: restaurant (create, search and delete), cuisine types (list) and order (add, remove, update, remove all, complete, list content, list previous)

Mahshid Shams contributed to this project by:

1. Participating in the creation of the ER schema database
2. Designing the user experience and creating a mockup for the figma
3. Designing the mockups for the restaurant and the dish, as well as the admin and manager's view of the figma.
4. Implementing the listUserDAO
5. Implementing the listUser Rest API
6. Adding real data to the mockups
7. Writing the documentation of the mockup for Restaurant manager view
8. Writing the documentation of the mockup for Dish manager view

Nicola Ursino contributed to this project by:

1. Helping in the ER schema development for the database.
2. Implementing the logging functionality.
3. Working on the application mock up, more specifically on the user profile and the search bars.
4. Creating an initial version of the java classes for each db entity.
5. Implementing the SearchDishesByRestaurantByNameByDiet REST API and DAO.
6. Implementing the SearchRestaurantByNameByCuisine REST API and DAO.
7. Implementing the UpdateUser REST API and DAO.
8. Implementing the ListUserByEmail REST API and DAO.
9. Implementing the ListCuisineTypes REST API and DAO useful for the advanced search.
10. Implementing all (except UpdateUser that had already been implemented by Daniel Carlesso) the SQL functions in *festival-query.sql* for the DAOs above.
11. Implementing the initial organization of the REST error codes into a java class (RestErrorCodes.java).
12. Rebuilding the RestDispatcherServlet.java from scratch.
13. Writing the User View section for the presentation logic layer.
14. Writing the documentation of the REST Error Codes for the business logic layer.
15. Writing the documentation of the REST API details for: dish (create, search and delete) and user (register, create, update, login, logout, delete, list, search, list cards).

7 Running the application

The application is designed to run in a docker environment. Please refer to README.md file in the repository for more detailed information about it (especially the last section for compiling and running the application without docker).