

Web Applications, A.Y. 2020/2021
Master Degree in Computer Engineering
Master Degree in ICT for Internet and Multimedia

Homework 1 – Server-side Design and Development

Submission date: 23 April 2021

Last Name	First Name	Badge Number
Friso	Luca	1236455
Mjama	Anass	1234003
Dallapellegrina	Elia	1220045
Grimaldi	Alberto	2026704
Golmohammadi	Pargol	2005932
Andarzian	Niloofer	2005864

Objectives

The purpose of the application is to allow a user to purchase a tailor-made garment. This garment can be chosen from a wide range of products available from the tailor's shop, or the final product can be created from scratch according to requirements. It is also possible to book an appointment for a repair of any tailor-made product. It will also be possible for the shop manager, and his/her employees, to manage orders received from customers.

Main functionalities

For this purpose, we developed a webapp consisting of:

- A **homepage**, containing a login button and two boxes to access the following pages:
 - o **Custom creation**: this page links to another page where you can choose between:
 - **Create from scratch**: allows you to create a garment according to your needs and to receive a response from the shop via email or a call.
 - **Book an appointment for a restoration**: allows you to book an appointment in the shop to have an item repaired or for any other need.
 - o **Shop page**: on this page you will be able to purchase products already finished by the tailor, with the possibility of choosing size and color where available.
- After logging in, the shop manager, or his co-workers, will be redirected to a page allowing them to manage the orders received, check for any appointments booked by customers, check the details of a particular customer, check for any custom products required and for the manager it will also be possible to manage their employees by adding, updating or deleting a profile.

The employee management privilege of the manager will be implemented during the construction of the database, and each time the manager enters a new employee it will be marked in the database with the role of "employee", thus only allowing access to order management. However, the possibility is left for a manager to insert another manager.

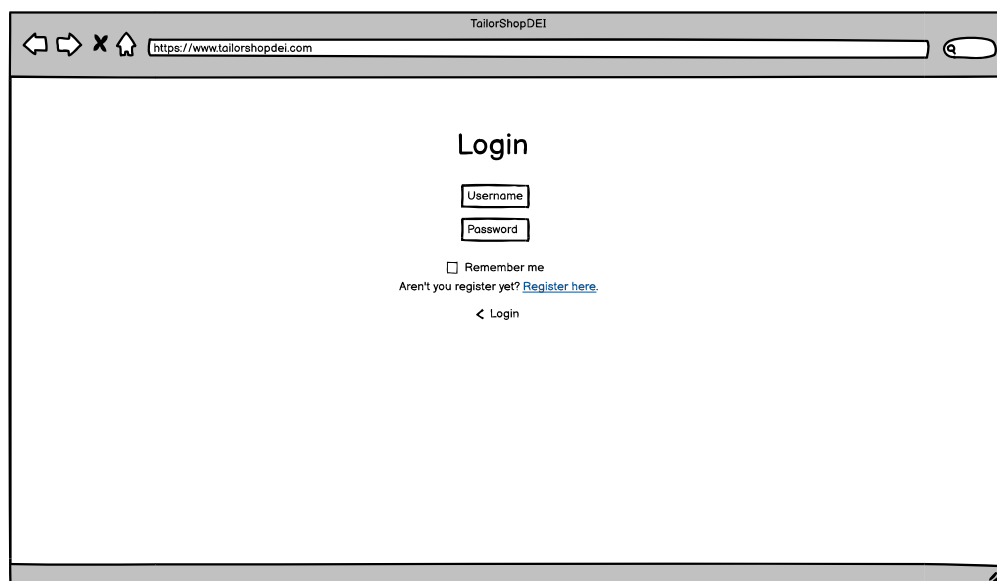
Presentation Logic Layer

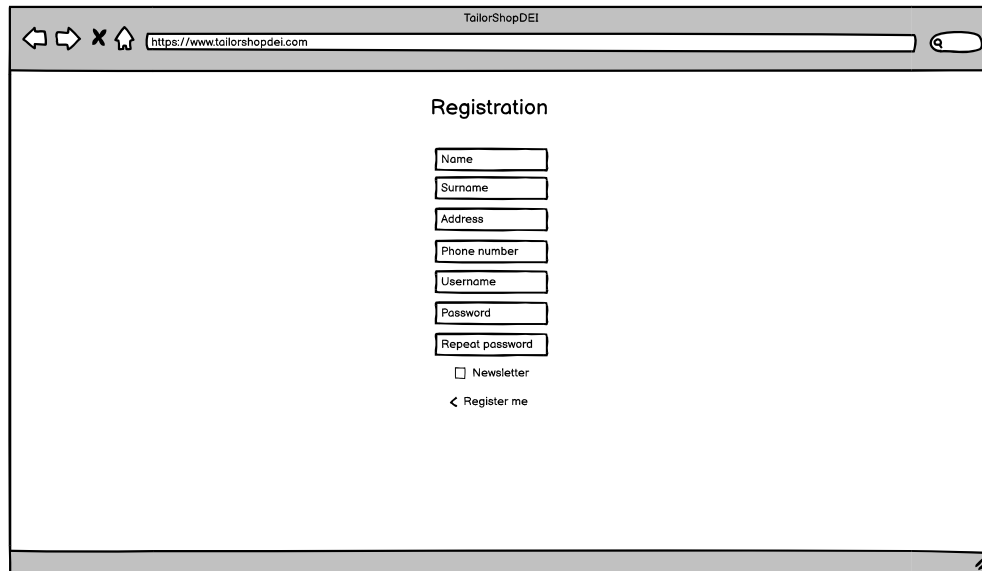
The project is divided into the following pages:

- Homepage: contains references to the shop and the custom creations. developed in jsp.
- Create from scratch: allows the user to create their own product. Developed in JSP with javascript support.
- Book an appointment: allows the user to book an appointment in the shop. Developed in REST with javascript support.
- Login page: allows the user to login to the application. Developed in jsp with javascript support.
- Registration page: allows the user to register to the web app. Developed in jsp with javascript support.
- Order management: allows the manager and employees to manage orders, whether it is an order for an existing product in the shop or an order for a new product, placed by customers. Developed in JSP with javascript support.
- Insertion of a new employee: allows the manager to insert, modify, delete the profile of an employee. Developed in JSP with the support of javascript.
- Insert of a new order: allows the customer to place an order for a product in the shop. Developed in JSP with the support of javascript.
- Insert of a new custom product: allows the customer to enter a new custom product they want and be contacted by the shop as soon as possible.

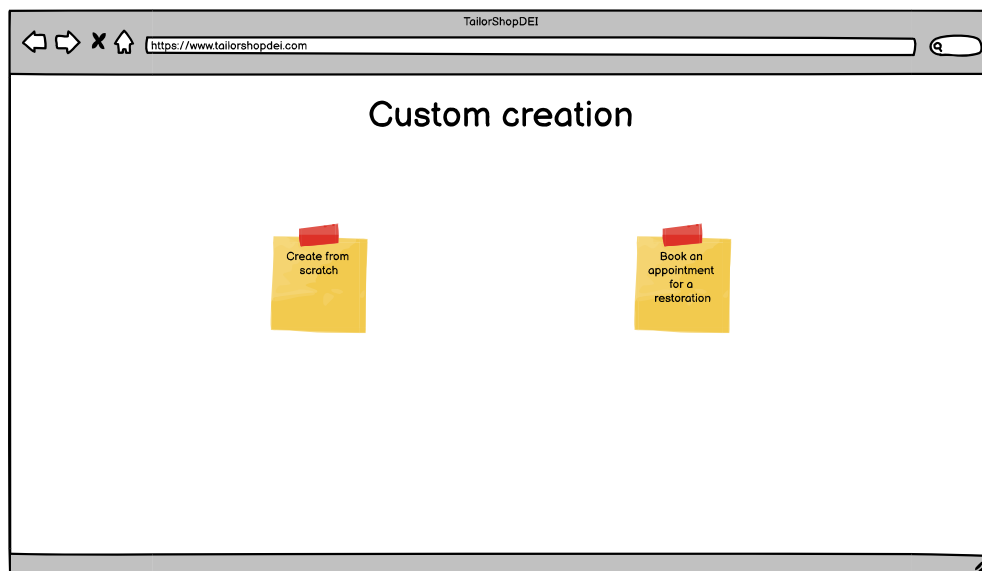
Home Page (Interface Mockup)

The homepage allows to choose which section of the app, shop or custom creation, to move to. It allows to login, access the shop or the custom creation section if the user is a customer. Otherwise, if the user trying to log in is a manager or an employee, he/she will be redirected to the page designated for him/her. A non-logged-in user can still visit the shop but will be redirected to the login page before being able to complete an order.





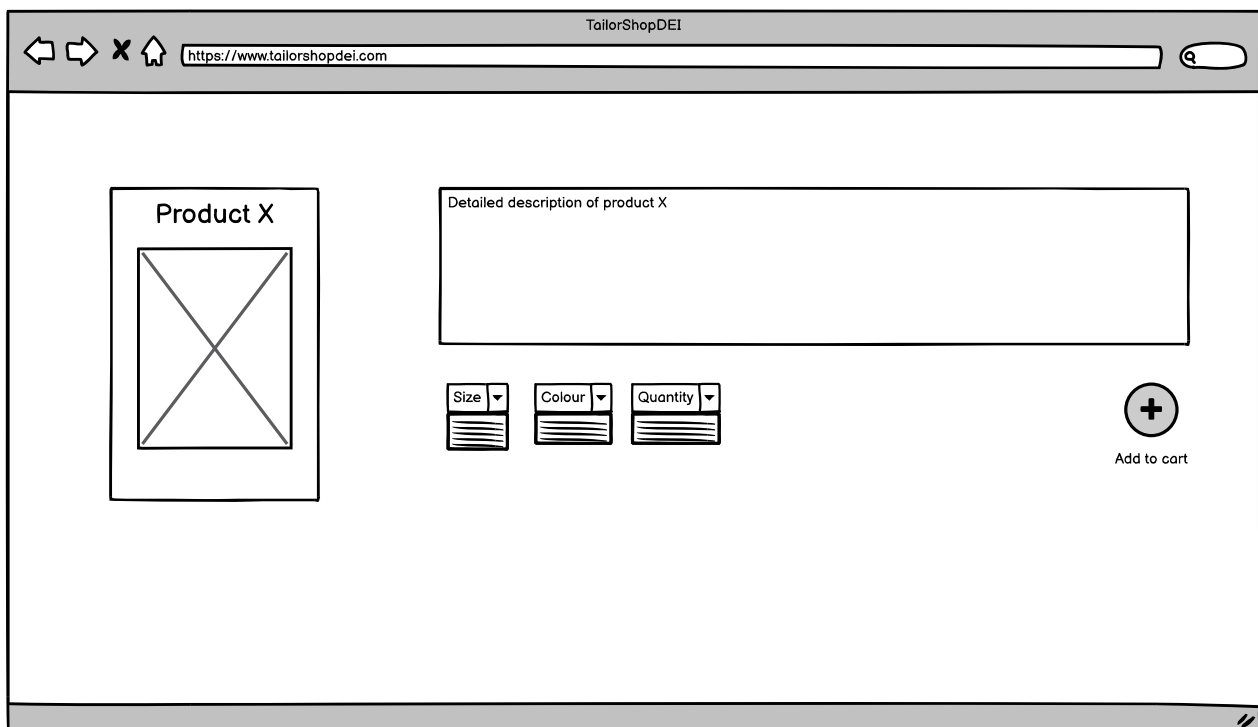
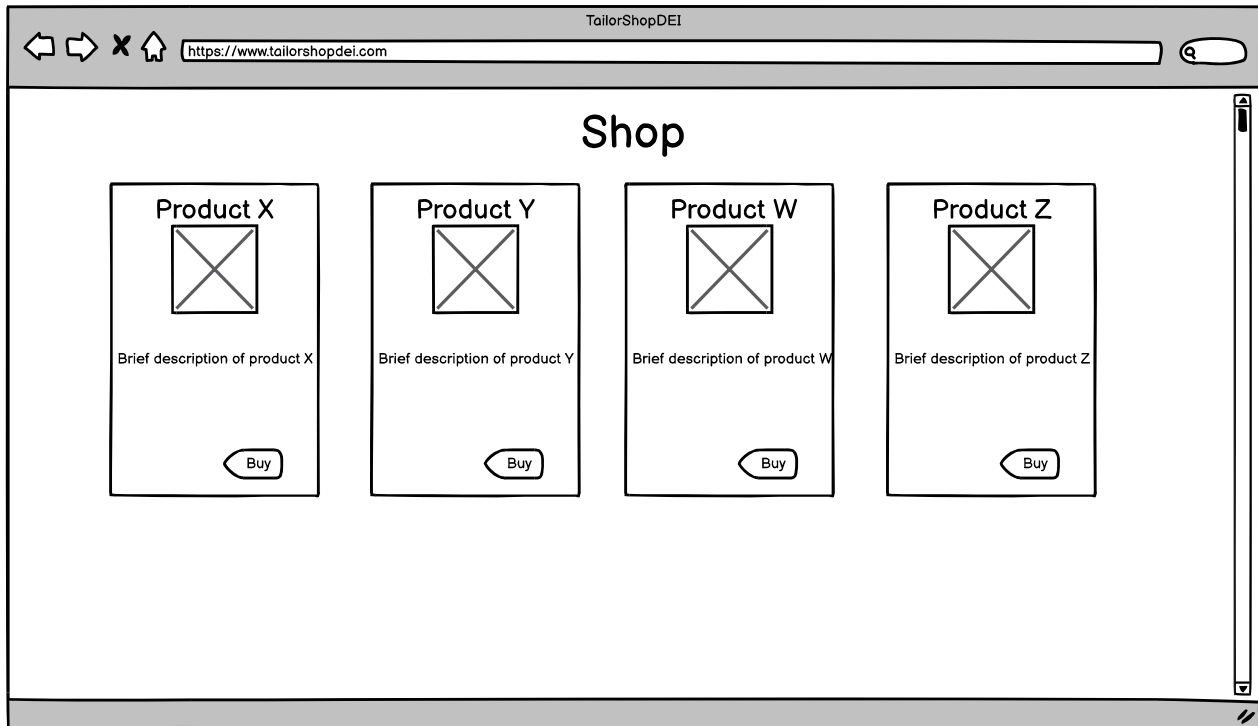
A browser window titled "TailorShopDEI" with the URL "https://www.tailorshopdei.com". The page displays a "Registration" form with the following fields: Name, Surname, Address, Phone number, Username, Password, and Repeat password. Below the fields is a checkbox for "Newsletter" and a "Register me" button.



A browser window titled "TailorShopDEI" with the URL "https://www.tailorshopdei.com". The page displays a "Custom creation" section with two yellow sticky-note-like buttons: "Create from scratch" and "Book an appointment for a restoration".

Shop Page (Interface Mockup)

The shop page is made up of several cards showing the product image and a brief description. A button allows to access the page dedicated to the selected product, allowing to choose the color, size and other details and finally complete the purchase.



Create from scratch Page (Interface Mockup)

It allows to create a new product from scratch via a form where the customer can enter some characteristics of the product he/she would like to be produced. If the entry is successful, he/she will be redirected to a page that allows the customer to return to the homepage.

The mockup shows a web browser window titled 'TailorShopDEI' with the URL 'https://www.tailorshopdei.com'. The page content is titled 'Create your own personalised suit'. On the left, there are five labels with corresponding dropdown menus: 'Choose the type of product' (Type), 'Choose your size' (Size), 'Choose the material' (Material), 'Choose the colour' (Colour), and 'Choose the motif' (Motif). Each dropdown menu has a small icon of a suit. To the right of these is a large text area labeled 'Add here some notes'. Below the dropdowns, there is a 'Book an appointment' section with a date input field (//) and a calendar icon, followed by the text 'or buy your custom creation' and an 'Add to cart' button. The browser window has standard navigation buttons (back, forward, home, search) and a search bar.

Book an appointment Page (Interface Mockup)

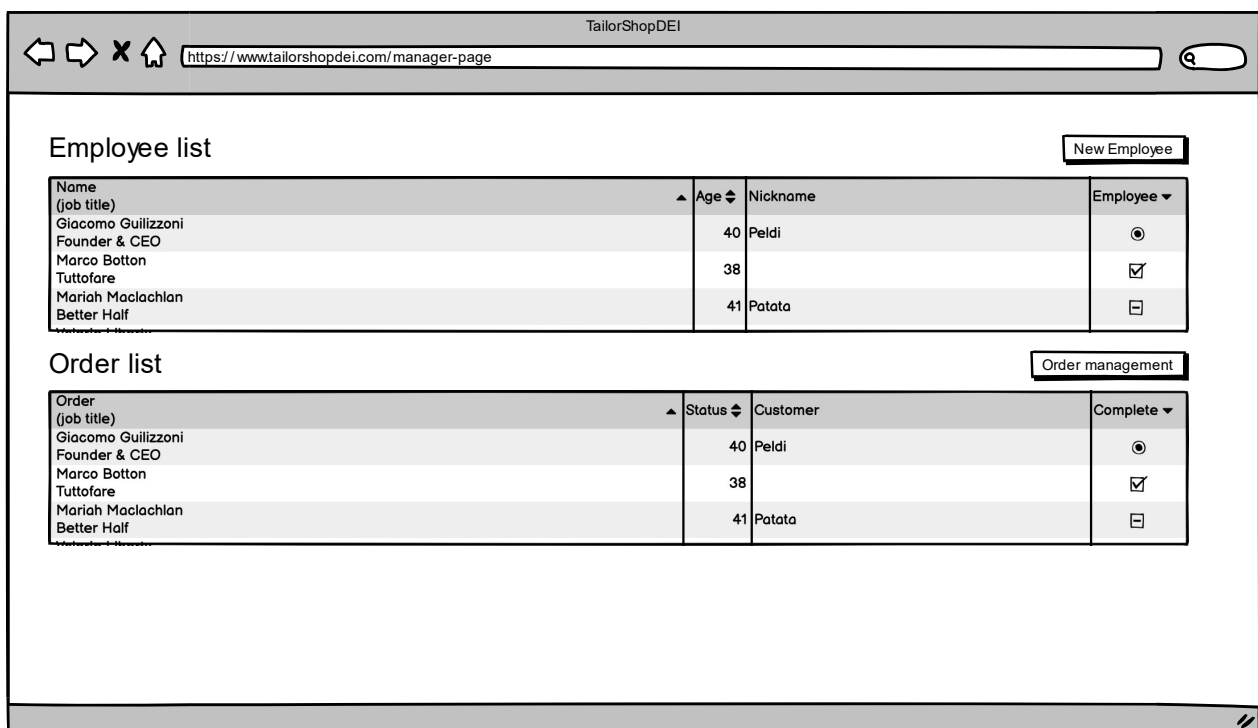
It allows the customer to book an appointment via a form where it is possible to select the date and the hour on which to go to the shop. It allows also to insert a brief description of what his/her needs are.

The mockup shows a web browser window titled 'TailorShopDEI' with the URL 'https://www.tailorshopdei.com'. The page content is titled 'Book an appointment for a restoration'. It features a form with five input fields: 'Name', 'Surname', 'Email', and 'Phone'. Below these is a 'Select a date' section with a date input field (//) and a calendar icon. At the bottom, there is a large text area labeled 'Describe your needs' and a 'Book' button. The browser window has standard navigation buttons (back, forward, home, search) and a search bar.

Manager Page (Interface Mockup)

After logging in a manager is redirected to his management page where he/she can choose which operation to perform:

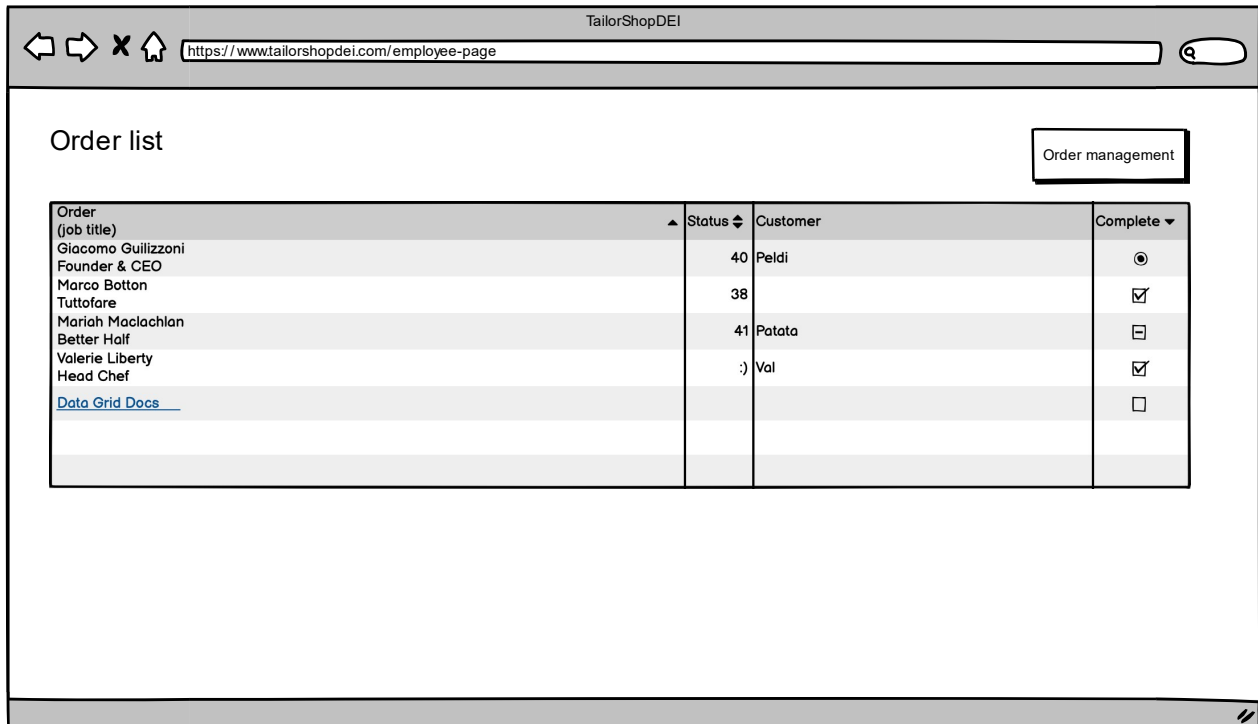
- insert a new employee: this is made possible by an insertion form.
- manage employee: in this section the manager can view all the details about an employee, modify some details or even delete it from the database.
- order management: in this section the manager can view the various orders placed by customers, change their status or delete them if the order has already arrived and the customer is satisfied.
- custom product management: in this page the manager can manage the requests for the creation of a custom product made by various users.
- appointment management: In this section the manager can check the various appointments that have been booked and manage them accordingly by marking any appointments that have already been made.



In this mockup example we have reported only the "Employee list" and the "Order list" because the other features described before will look like the once reported in the image.

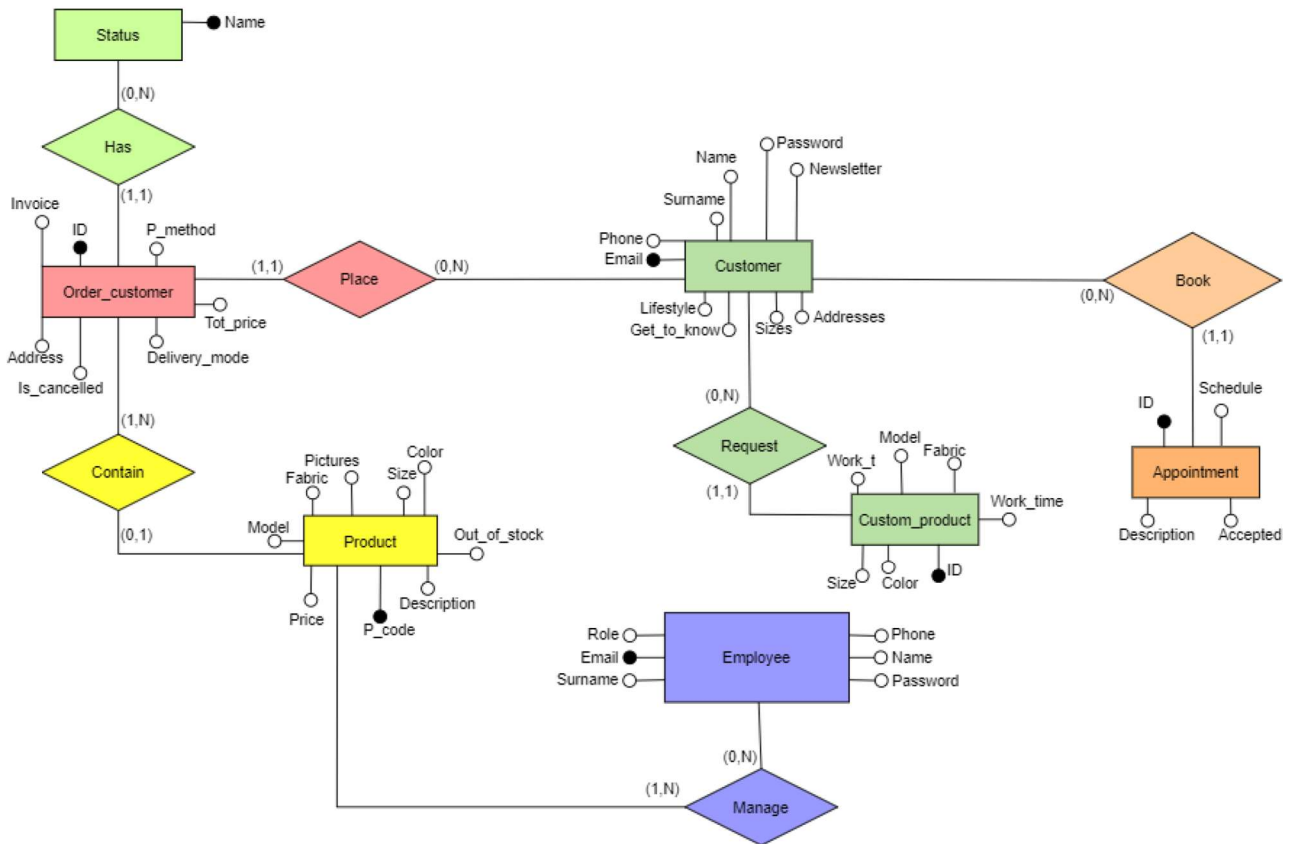
Employee Page (Interface Mockup)

This page is very similar to the manager page just for the difference that the employee cannot insert a new employee.



Data Logic Layer

Entity-Relationship Schema



The entity-relationship contains 8 main entities:

- **Customer** : describes who orders the product. Each customer has, as a primary key, their email, which is of type char with variable length (up to 64 characters). For each customer we also register their name, surname (of type varchar), phone (varchar), newsletter that is of type Boolean to indicate if the customer is subscribed or not to the newsletter, get_to_know where the customer specifies how got to know the tailor's shop, password, sizes (varchar), lifestyles (varchar), addresses (is an array of varchar). Note that the password is hashed through md5 before storing it. Each customer can place an order (determining a 1-N relation with order_customer), book an appointment (determining a 1-N relation with appointment) or send a request for a customized product (determining a 1-N relation with custom_product).
- **Status** : each order is associated with a status identified by a name, which is of type *orderstatus*, a custom enumeration. The relationship "Has", between status and order_customer, is of type 1-N.
- **Order_customer** : is the order placed by a customer. Its primary key, ID is of type UUID. There are also six additional attributes: the total price, that is the sum of the prices of the products contained in that order; the address of the order; the payment method which is of type *paymentmethod* (custom enumeration); the delivery mode of type *deliverymode*, a custom enumeration data type; is_cancelled is of type Boolean that indicates whether the order is cancelled or not; finally, the invoice of type varchar. The customer who places the order and the status associated with the order are therefore stored as external keys directly on the order entity instances.
- **Employee** : is the Tailor's shop worker. The primary key is the email followed by other attributes like name, surname, phone number password and role (of type varchar). Each employee has his/her own account, through which can perform all the basic functionalities of the system.

Managers are also allowed to register or delete an employee account. An employee can also manage a product, and this is represented by the N-N “Manage” relationship.

- **Product** : product of the tailor’s shop. It is identified uniquely based on an UUID variable, *p_code*. When a customer purchases some products, they are contained in that customer’s order (determining a relation N-1 with the orders). Other attributes are model, fabric and size, represented through varchars, price of type numeric, the pictures of the products of type bytea array, the color of type *color*, a custom enumeration, the textual description of the product and *out_of_stock* of type boolean.
- **Appointment** : when a customer wants to go directly to the tailor's shop to propose particular requests maybe regarding the creation / modification of a suit or simply to request a consultation, he/she can request an appointment. The appointment entity has as primary key an integer with autoincrement and other attributes like the schedule of the appointment, of type timestamp, the textual description and the Boolean attribute “accepted” which indicates whether the appointment was accepted or not.
- **Custom_product** : each customer can decide to request a new customized product or simply a repair for an item of clothing (determining a relationship N-1 with customer). The primary key of a custom product is of type UUID. Then there is a model, fabric and size of type varchar, *work_time* of type smallint which provides an upper bound of hours of work within the customer would like the product to be finished, the color, the work type which specify whether the request is for the creation of a new product or for the tailoring of an item. Each custom product is linked to a specific customer and thus it contains as a foreign key the email of the customer.

Other Information

There are 4 custom enumerations:

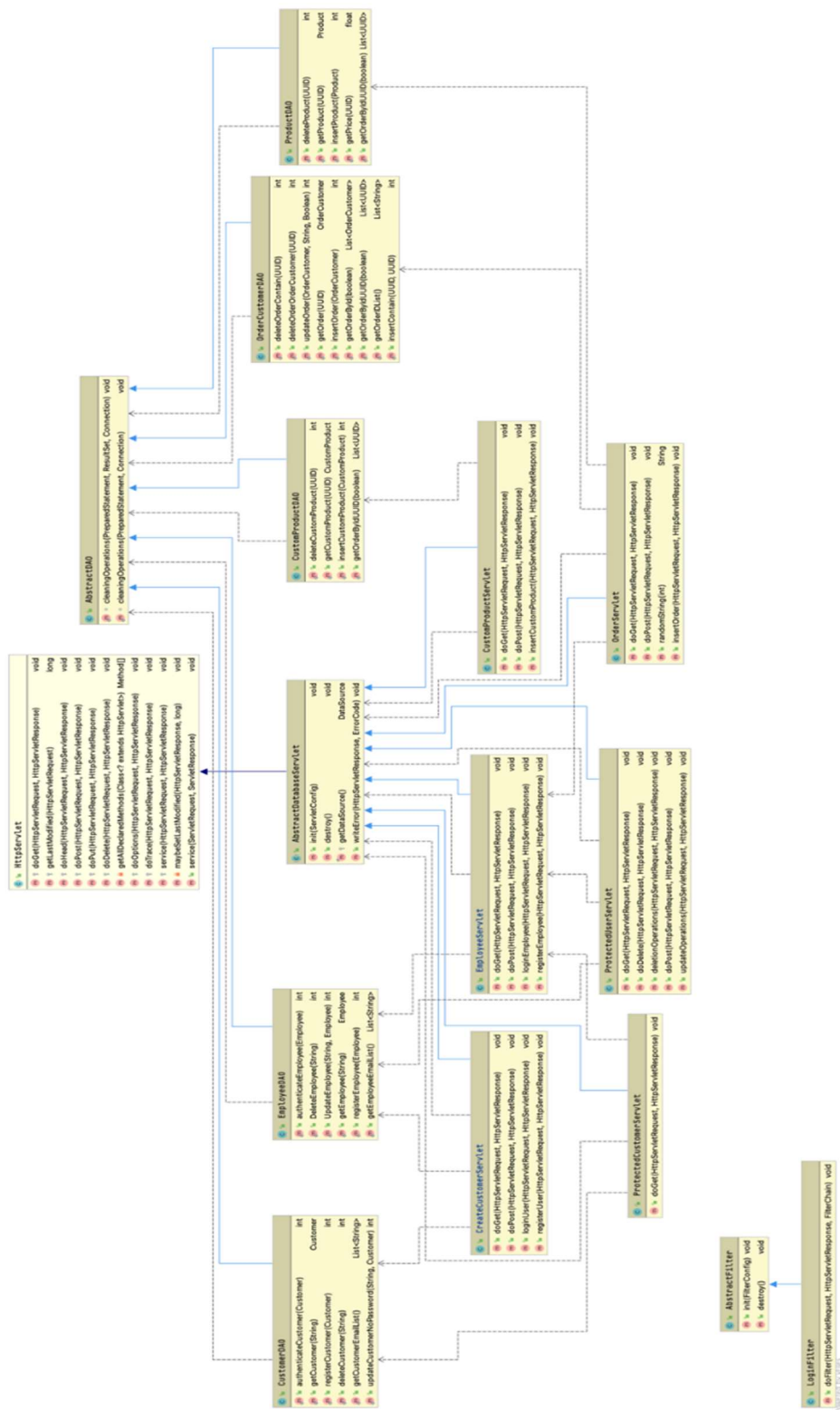
- *paymentmethod* indicates the different type of payments allowed;
- *orderstatus* identifies six different status of the order like 'Pending', 'Accepted', 'Processing' and so on;
- *deliverymode* which indicate the modality of delivery that a customer can choose ('Home and 'Shop');
- *worktype* to distinguish if the customer wants to request a repair or a customized product ('From scratch', 'Tailor').

When a customer intends to create a customized garment or wants to request a repair, he/she can formalize a request for a custom product by appropriately modifying the *work_type* attribute on the web site. In this way the customer will be contacted by the tailor’s shop who will provide more detailed information based on the features of the request made. In the case in which the customer already knows the detailed information about his/her request (like for example the price), he/she could directly book an appointment.

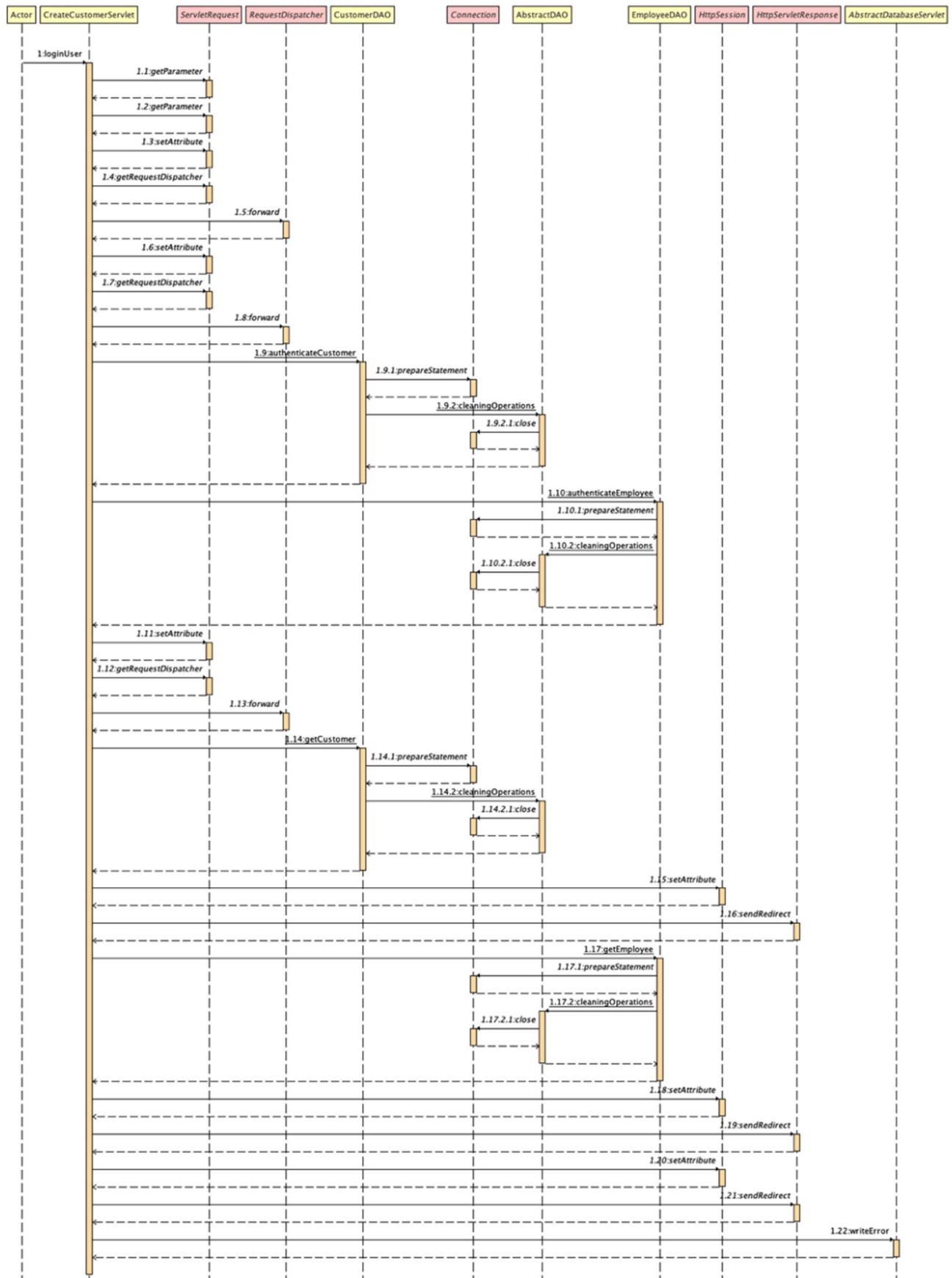
If a customer choose ‘Home’ has delivery mode, he/she will be contacted by the tailor’s shop when the item is ready

It is possible to interact with the most of entities (insertion, deletion and update) directly via interface.

Class diagram



Sequence Diagram



The sequence diagram for a user login is shown above. The actor, in our case a customer or an employee or a manager, makes a POST request to the web server specifying user/login (this step is clearly visible in the *CreateCustomerServlet.java* file) and entering the username and password. The web server instantiates *CreatecustomerServlet* and calls the *doPost* method, passing *HttpServletRequest* and *HttpServletResponse*. *CreateCustomerServlet* parses the request and, having recognized that it is a login attempt, calls the *loginUser* method which checks that the email and password belong either to a customer or to a manager or employee. If the credentials entered are incorrect, the user is sent back to the login page and is notified of an error specifying whether the email or password was incorrect. If the credentials are correct, two new objects are created, a *Customer* object and an *Employee* object and both are passed to the corresponding *authenticateCustomer* and *authenticateEmployee* methods in *CustomerDAO* and *EmployeeDAO* respectively, which contact the Database server to check whether the email and password are present in the Customer or Employee tables respectively. If the credentials belong to a customer, *CreateCustomerServlet* obtains the session from *HttpServletRequest* and sets the email parameter in the session, it is then sent back to *homepage.jsp* from which it can continue browsing the web app. If, on the other hand, the credentials belong to an employee, a further check is performed by calling the *getRole* method in *EmployeeDAO*. If the resulting role is "manager" the user is sent back to the page dedicated to him "*manager-page.jsp*", otherwise if the resulting role is "employee" the user is sent back to the page dedicated to him "*employee-page.jsp*".

REST API Summary

In our project we use the REST paradigm to implement an Appointment resource. In particular, some endpoints require a {id} identifier to identify a specific appointment, while other endpoints require no additional parameters in the URI.

URI	Method	Description
rest/appointment	GET	Returns the list of all appointments available in the database
rest/appointment/{id}	GET	Allows to get information on the appointment identified by the id {id}
rest/appointment/{id}	PUT	Allows to update the information about the appointment identified by the id {id}
rest/appointment/{id}	POST	Allows to insert a new appointment with id {id}
rest/appointment/{id}	DELETE	Allows to delete the appointment identified by the id {id}
rest/appointment/pending	GET	Returns the list of all appointments that are still waiting to be accepted

REST API Error Codes

Here we can see the list of errors that the application might arise. The error codes are numbered from -102 and are divided in 3 categories: from -102 to -108 there are errors related to the login/registration phase, from -113 to -129 there are errors related to the REST part of the application and finally -200, -500 and -999 are generic errors that refer to not allowed or unknown methods/operations or generic internal errors like crashes or exceptions.

Error Code	HTTP Status Code	Description
-102	BAD_REQUEST	The user tried to register without specifying all the details.
-103	BAD_REQUEST	The user tried to login without specifying the Email.
-104	BAD_REQUEST	The user tried to login without specifying the password.
-105	BAD_REQUEST	User submitted wrong credentials in the login form.
-106	NOT_FOUND	User to be deleted not found.
-107	CONFLICT	Different passwords when repeating the password in the registration phase.
-108	CONFLICT	Email already used.
-113	BAD_REQUEST	Wrong rest request format.
-120	BAD_REQUEST	The input json is in the wrong format.
-126	NOT_FOUND	Appointment to be selected not found.
-127	CONFLICT	Appointment already exists.
-128	CONTENT	Date not valid.
-129	NOT_FOUND	Appointment does not exist.
-200	BAD_REQUEST	Operation unknown.
-500	METHOD_NOT_ALLOWED	The method is not allowed.
-999	INTERNAL_SERVER_ERROR	There was an internal error that the server was not able to manage.

REST API Details

We report here 3 different resource types that our web application handles during its functioning.

<Appointment List>

This resource contains a list of all appointments available in the database.

- URL
rest/appointment
- Method
GET
- URL Params

No parameters are required in the url

- **Data Params**

No data is passed when requiring this method

- **Success Response**

If success, the servlet returns a JSON object containing the list of all appointments in the database

Code: 200

Content:

```
{"data":
  {"appointments-list":
    [
      {
        "schedule":"2010-02-07 15:00:00.0",
        "accepted":false,
        "id":"75d0f606-b5aa-4852-b4dc-e159a1560831",
        "customer":paolorossi@gmail.com},
      {
        "schedule":"2010-05-09 10:00:00.0",
        "accepted":false,
        "id":"fff0a2cf-c05a-47ed-bdb9-1d5f2a40afef",
        "customer":tiziocaio@libero.it},
      ...
    ]
  }
}
```

- **Error Response**

Code: 126 APPOINTMENT_NOT_FOUND

Content: {code : -126, description : "Appointment not found."}

Code: 999 INTERNAL_ERROR

Content: {code : -999, description : "Internal error."}

<Update Appointment>

This endpoint allows to insert, update, retrieve or delete an appointment, in this case we present only the updating operation.

- URL
rest/appointment/{id}
- Method
PUT
- URL Params
Required:
 - id = {uuid}
The identifier of the appointment
Optional:
 - none
- Data Params
Required:
 - id = {uuid}
Id that identifies the appointment
 - schedule = {timestamp}
Date and hour of the appointment
 - accepted = {boolean}
A flag to indicate if the appointment has been accepted by the shop
 - customer = {tailor_feda.emailaddress}
The email address of the customer
Optional:
 - description = {string}
The description of the appointment
- Success Response

Code: 200
Content: {error : false, message: "Appointment updated correctly"}
- Error Response

Code: 128 DATE_NOT_VALID
Content: {code : -128, description : "Date not valid."}
Code: 126 APPOINTMENT_NOT_FOUND
Content: {code : -126, description : "Appointment not found."}
Code: 999 INTERNAL_ERROR
Content: {code : -999, description : "Internal error."}
Code: 120 BADLY_FORMATTED_JSON
Content: {code : -120, description : "The input json is in the wrong format."}