Graphical user interface, text, application, email

Description automatically generated

Deluxe Perfumes

DOCUMENTATIE DE PROIECT

*Negrut Ciprian-Petrica*

*Grupa: 30235*

*An universitar:2022-2023*

*Semestrul: 2*

Table of Contents

[Deliverable 1 3](#_Toc64843130)

[Project Specification 3](#_Toc64843131)

[Functional Requirements 3](#_Toc64843132)

[Use Case Model 3](#_Toc64843133)

[Use Cases Identification 3](#_Toc64843134)

[UML Use Case Diagrams 3](#_Toc64843135)

[Supplementary Specification 3](#_Toc64843136)

[Non-functional Requirements 3](#_Toc64843137)

[Design Constraints 3](#_Toc64843138)

[Glossary 3](#_Toc64843139)

[Deliverable 2 3](#_Toc64843140)

[Domain Model 3](#_Toc64843141)

[Architectural Design 4](#_Toc64843142)

[Conceptual Architecture 4](#_Toc64843143)

[Package Design 4](#_Toc64843144)

[Component and Deployment Diagram 4](#_Toc64843145)

[Deliverable 3 4](#_Toc64843146)

[Design Model 4](#_Toc64843147)

[Dynamic Behavior 4](#_Toc64843148)

[Class Diagram 4](#_Toc64843149)

[Data Model 4](#_Toc64843150)

[System Testing 4](#_Toc64843151)

[Future Improvements 4](#_Toc64843152)

[Conclusion 4](#_Toc64843153)

[Bibliography 4](#_Toc64843154)

# Deliverable 1

## Project Specification

This project aims to develop a Deluxe Perfumes web application that can be used to order any perfume the user wants. Only, the Admin cand add/delete/update a perfume and user can only order it. The application also includes features for filtering perfume by categories and price.

## Functional Requirements

1. User registration and authentication
2. User create an order
3. Filtering products by category
4. Filtering products by price
5. Admin can create a new product
6. Admin can delete a product
7. Admin can update a product

## Use Case Model

### Use Cases Identification

Use-Case: Register a new User

Level: User global

Primary Actor: Unregistred User

Main success scenario:

1. Unregistered User access the registration page.
2. Unregistered User fill out the registration form.
3. Unregistered User submits the form.
4. System validates the information and creates a new user account.

Use-Case: Login User

Level: User global

Primary Actor: Registered User

Main success scenario:

1. User access the login page.
2. User fill out the login form.
3. User submits the form.
4. System validates the information and redirects User to home page.

Use-Case: Create Order

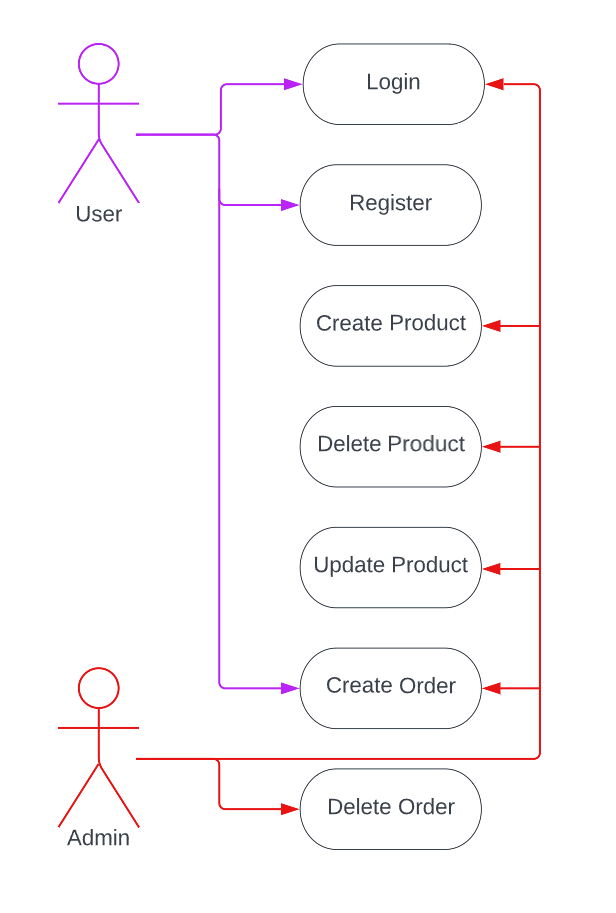
Level: User global

Primary Actor: Logged User

Main success scenario:

1. User access the create order page.
2. User fill out the order form.
3. User submits the form.
4. System validates the information and notiy the User about the order created.

### UML Use Case Diagrams



## Supplementary Specification

### Non-functional Requirements

1. Usability: The application should have an intuitive user interface, making it easy for users to navigate and find the information they need.
2. Scalability: The system should be able to handle a growing number of users and job listings without significant performance degradation.
3. Security: User data must be protected, and sensitive information like passwords should be securely stored using encryption and hashing techniques.
4. Responsiveness: The application should provide quick response times and load times, ensuring a smooth user experience.

### Design Constraints

1. The project must be developed using Java and Spring Boot as the back-end framework.
2. Front-end development: JavaScript framework, Angular
3. The database should be designed using a relational database management system (MySQL).
4. The application should follow a RESTful API design.
5. Clean code, including proper documentation and testing.

## Glossary

Deluxe Perfume app: web application that allows users to order perfumes product.

Users are allow to register new account in app and they can create order with perfumes product which are available on site. Only the Admin can create and add new product in shop.

# Deliverable 2

## Domain Model

1. *Name:* BaseEntity

*Attributes:* id | createdDate | updatedDate | externalId

*Associations:* an abstract Entity

1. *Name:* Order

*Attributes:* identifier | username | orderCreatedMessage | orderPrice | perfumes + [ BaseEntity Attributes]

*Associations:* an order can have many perfumes

1. *Name:* Perfume

*Attributes:* name | category | identifier | description | price | order +

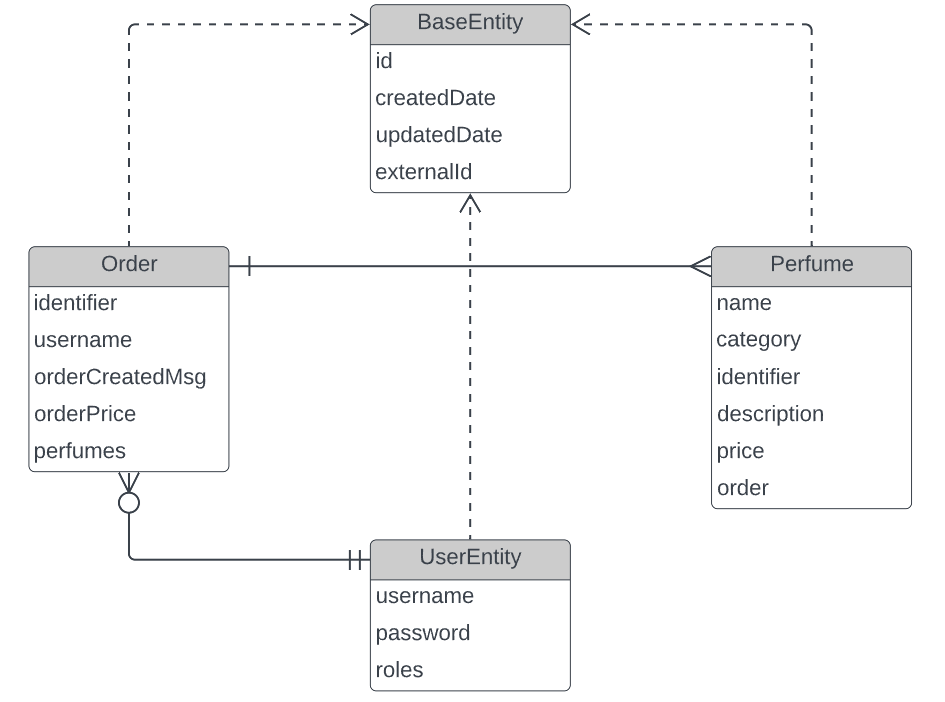
[ BaseEntity Attributes]

*Associations:* perfume product

1. *Name:* UserEntity

*Attributes:* username, password, roles + [ BaseEntity Attributes]

*Associations:* a person who uses the delxue-perfumes app



## Architectural Design

### Conceptual Architecture

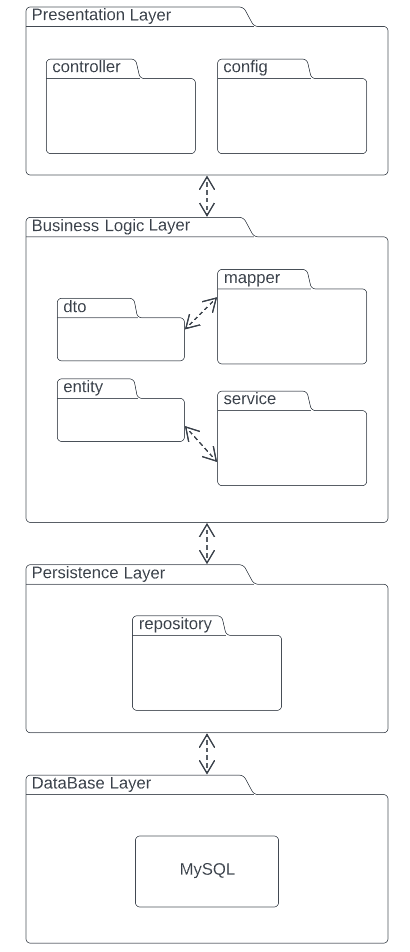
The architectural style used for the system is REST (Representational State Transfer) and the architectural pattern is MVC (Model-View-Controller).

The REST architectural style is well suited for designing web-based systems that are scalable, flexible, and easy to integrate with other systems and services. REST defines a set f constraints and principles for creating stateless, client-server systems that communicate using a uniform interface.

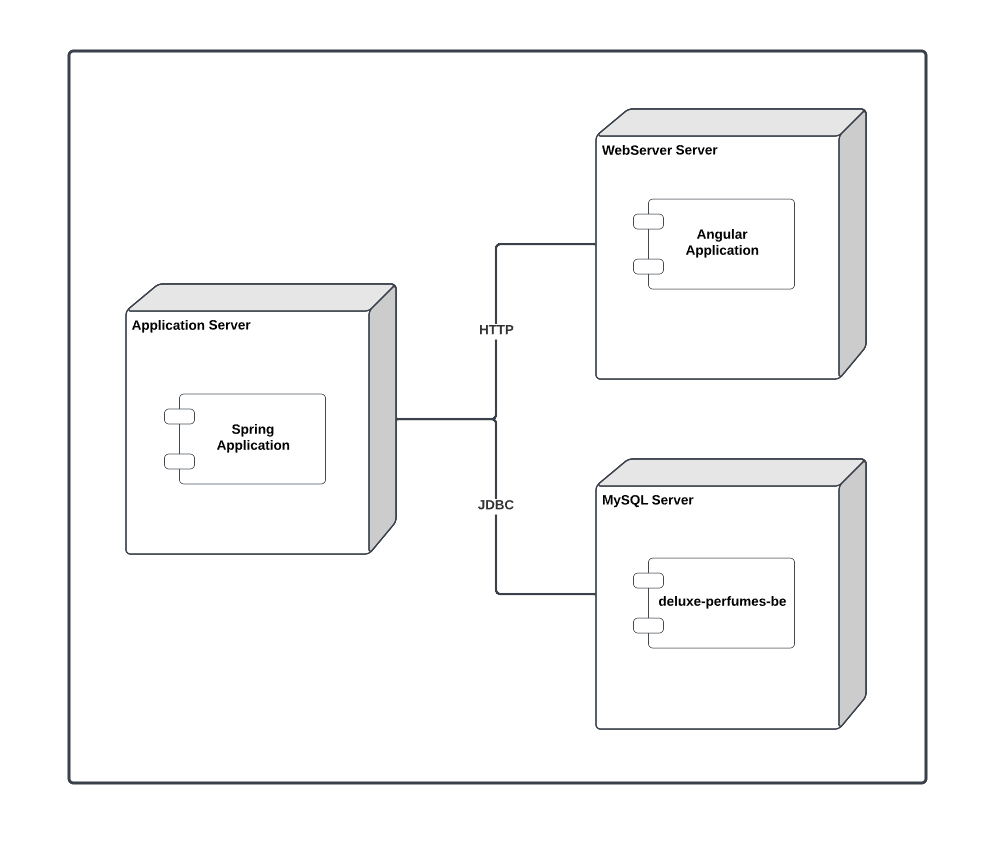
The MVC pattern is a widely used pattern in web development that separates the application into three main components: the Model, the View and the Controller. The Model represents the data and the business logic of the application, the View represents the user interface, and the Controller handles user input and updates the Model and View accordingly.

In the case of my application, the REST architecture is used to create a set of resources that can be accessed and manipulated using HTTP requests. For example, the system has resources for tracking the progress made by users. Clients can access these resources usgin HTTP requests, and the server can respond with the appropriate representations of the resources.

### Package Design



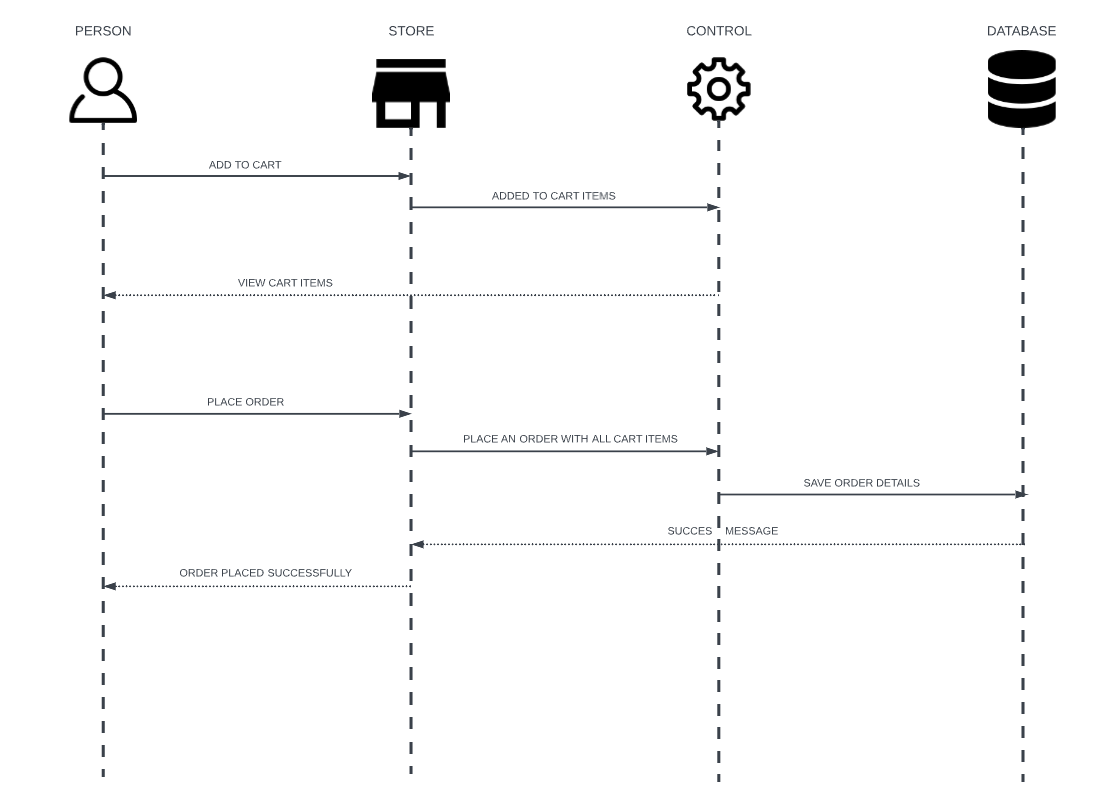
### Component and Deployment Diagram



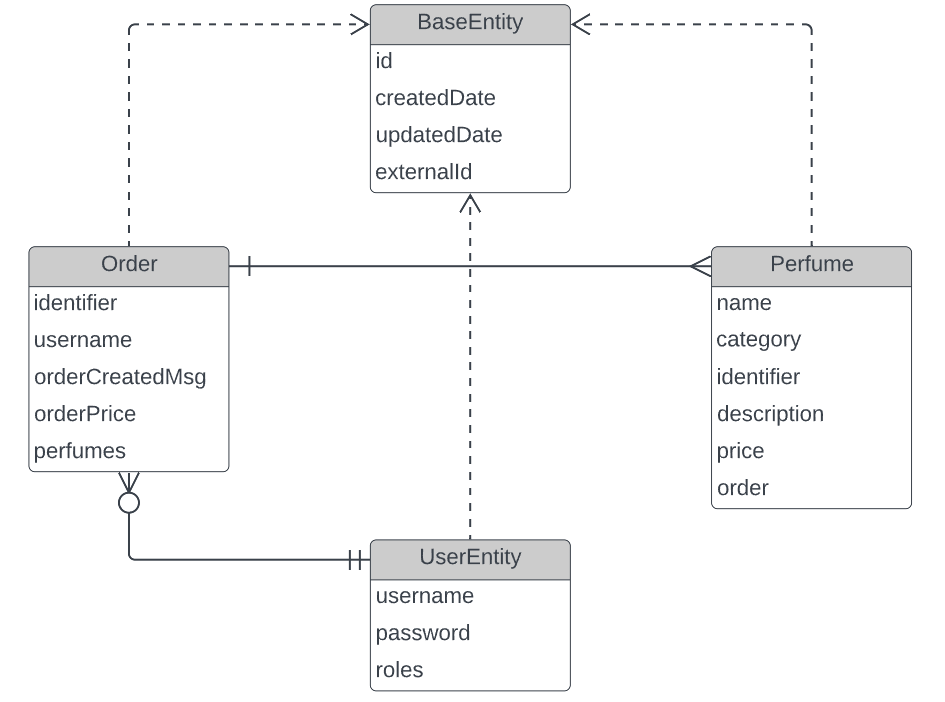
# Deliverable 3

## Design Model

### Dynamic Behavior



### Class Diagram



GoF Design Patterns are divided into three categories:

1. *Creational:* The design patterns that deal with the creation of an object.
2. *Structural:* The design patterns in this category deals with the class structure such as Inheritance and Composition.
3. *Behavioral:* This type of design patterns provide solution for the better interaction between objects, how to provide lose coupling, and flexibility to extend easily in future.

I used Observer pattern because i needed to receive an message when an order was placed.

## Data Model

Entities:

1. User - represent users who access the online store and place orders.

* Username
* Password
* Role

1. Product - represent the products available in the online store.

* Name
* Category
* Identifier
* Description
* Price
* Promo
* outOfStock
* Order - @ManyToOne relation

1. Order - represents an order placed by a user.

* Identifier
* Username
* orderCreatedMessage
* orderPrice
* Perfumes - @OneToMany relation

1. BaseEntity - represents the base entity inherited by all classes

* Id
* createdDate
* updatedDate
* externalId

# System Testing

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

# Future Improvements

Here are some potential future improvements for a web application that has product, order, and user functionality:

1. Implement a recommendation system: A recommendation system could be used to suggest products to users based on their past orders or browsing history. This could help increase user engagement and drive more sales.
2. Enable user reviews and ratings: Allowing users to leave reviews and ratings on products they've purchased can help build trust with potential customers and improve the overall quality of the products offered.
3. Integrate a loyalty program: A loyalty program could incentivize users to make repeat purchases by offering discounts or other rewards for their continued business.
4. Offer multiple payment options: Providing users with a variety of payment options can make the checkout process more convenient and user-friendly. This could include options like credit/debit cards, PayPal, Apple Pay, Google Pay, etc.
5. Streamline the checkout process: Simplifying the checkout process can help reduce cart abandonment rates and increase conversions. This could include implementing a guest checkout option, reducing the number of steps required to complete a purchase, and optimizing the checkout form for mobile devices.
6. Implement a chatbot or live chat feature: A chatbot or live chat feature could be used to provide users with instant support and answer any questions they may have about the products or the ordering process.
7. Provide order tracking: Giving users the ability to track their orders can help build trust and reduce customer support inquiries.
8. Personalize the user experience: Personalizing the user experience based on their past orders or browsing history could help increase engagement and drive more sales. This could include displaying personalized product recommendations or offering personalized discounts.
9. Implement social media sharing: Adding social media sharing buttons to product pages can make it easier for users to share products with their friends and followers, potentially driving more traffic and sales to the site.

# Conclusion

In conclusion, working on a perfume web application project with a backend in Java Spring, frontend in Angular, and database MySQL, along with security using JWT token has been a challenging but rewarding experience. The use of these technologies and tools allowed for a robust and scalable application that could handle complex operations and support a large user base.

Java Spring provided a powerful backend framework with a wide range of functionalities that enabled developers to build complex business logic and integrate with various third-party APIs. Angular, on the other hand, offered a responsive and dynamic frontend that could provide a seamless user experience.

The integration with MySQL database allowed for efficient storage and retrieval of data, while the implementation of JWT token-based security ensured the protection of sensitive user information and restricted access to authorized personnel only.

Overall, the perfume web application project provided an excellent opportunity for developers to showcase their skills and expertise in creating a high-quality, user-friendly, and secure application that met the needs and expectations of customers. The use of these modern and sophisticated technologies and tools has resulted in a robust, scalable, and efficient system that is capable of handling a significant volume of user traffic and transactions.

# Bibliography

<https://support.bigcommerce.com/s/question/0D51B00005YmubDSAR/can-i-edit-the-htmlcss-to-make-it-show-when-a-product-is-out-of-stock?language=en_US>

<https://www.google.com/search?q=how+to+disable+color+for+button+in+disabled&oq=how+to+disable+color+for+button+in+disabled&aqs=chrome..69i57j33i160l2.12672j0j7&sourceid=chrome&ie=UTF-8>

<https://www.google.com/search?q=how+to+set+disable+button+color&oq=how+to+set+disable+button+color+&aqs=chrome..69i57j33l2j33i160l3.10614j0j7&sourceid=chrome&ie=UTF-8>

<https://www.databasestar.com/sql-boolean-data-type/>

<https://www.google.com/search?q=how+to+get+value+from+Subject%3Cnumber%3E&oq=how+to+get+value+from+Subject%3Cnumber%3E&aqs=chrome..69i57j33i160l2.9313j0j4&sourceid=chrome&ie=UTF-8>

<https://www.geeksforgeeks.org/how-to-include-one-css-file-in-another/>

<https://stackoverflow.com/questions/9433448/html-if-statement>

<https://www.tutorialsteacher.com/typescript/typescript-if-else>

<https://www.reddit.com/r/TheInsaneApp/comments/12dmiqj/typescript_project_with_source_code_build_a_heart/>

<https://www.reddit.com/r/Angular2/comments/g9nbpz/adding_add_to_favorite_button/>

<https://stackblitz.com/edit/favorite-component-example>

<https://stackoverflow.com/questions/10853763/how-to-make-td-cell-from-html-table-expand-for-the-row-to-fit-in-one-line>

<https://www.google.com/search?q=how+to+make+a+pop+up+windows+for+editing+product+in+html&oq=how+to+make+a+pop+up+windows+for+editing+product+in+html&aqs=chrome..69i57j33i160.11312j0j7&sourceid=chrome&ie=UTF-8>