GÖTÜR Project

Design

Version 1.0

Prepared By:

MESE Company

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 23/04/2022 | 1.0 | Draft 1 | MESE TEAM |

# **Design Structure**

The core of the design is the layered architecture pattern, which is formed by individual layers performing a specific role within the application such as presentation layer, business layer and within each layer application’s components are categorized into. GOTUR application layered architecture consists of presentation, business, persistence and database layers:

* Presentation layer is responsible for handling all user interface and browser communication logic.
* Business layer is responsible for executing specific business rules associated with the request.
* Persistence layer is responsible for serving as an intermediary between the business functions of the application and the data it stores in a database.
* Database layer is responsible for containing the database where the application’s data that need to be saved and retrieved are located.

# **Subsystems**

* Config Package

Responsible for performing web configurations.

* Controller Package

Responsible for handling requests.

* DAO Package

Responsible for saving and retrieving information from the database.

* Entity Package

Responsible for containing persistent domain objects.

* Form Package

Responsible for containing object representations of user interface windows.

* Model Package

Responsible for containing non-persistent, information objects.

* Pagination Package

Responsible for adjusting paginated windows (such as product list window).

* Service Package

Responsible for various service interfaces such as login, payment.

* Utils Package

Responsible for containing utility functions.

* Validator Package

Responsible for user interface validation.

* MySQL Database

Responsible for storing data.

* User Interface (UI)

Responsible for containing visual components.

# **Patterns**

## **Model View Controller (MVC)**

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. The Model component corresponds to all the data-related logic that the user works with, the View component is used for all the user interface (UI) logic of the application and the Controller act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.

# **Requirement Realizations**

## **Manage Products**

productList and shoppingCart components from the presentation layer use requests to fulfill operations. These operations can be retrieving and posting data. The requests are met by MainController from the business layer. MainController operates with ProductDAO from the persistence layer in order to retrieve or post information. ProductDAO queries the MySQL database from the database layer and retriever the wished information or modifies it. Afterwards, information retrieved by ProductDAO is passed to MainController and MainController fulfills the request realization and the UI is then updated accordingly. In this way, listing products, adding to cart, updating products added in the cart, removing products from the cart requests are realized that form the scope of Manage Products.

## **Manage Product Specifications**

product and productList components from the presentation layer use requests to fulfill operations. These operations can be retrieving and posting data. The requests are met by MainController and AdminController from the business layer. MainController and AdminController operates with ProductDAO from the persistence layer in order to retrieve or post information. ProductDAO queries the MySQL database from the database layer and retriever the wished information or modifies it. Afterwards, information retrieved by ProductDAO is passed to MainController or AdminController and MainController or AdminController fulfills the request realization and the UI is then updated accordingly. In this way, update product information, remove product from the store, add product to the store requests are realized that form the scope of Manage Product Specifications.