restaurant-system

Generated by Doxygen 1.15.0

Chapter 1

README

Black Plate: Restaurant Management System using Maven Java

About

The profession of managing a restaurant. It includes the principles of OOP and using Java function of planning, organizing, staffing, directing, developing an attitude in food and beverage control systems, and efficiently and effectively planning menus at profitable prices, taking into consideration constraints and others.

More info on the wiki

https://github.com/alu0101132617/restaurant-system/wiki

Tools used:

- Java Development Kit (JDK)
- Maven
- JUnit (for testing)

Database:

- MySql

Functionality:

- Increases operational efficiency.
- Helps the restaurant manager to manage the restaurant more effectively and efficiently by computerizing Meal Ordering, Cart, and Restaurant Management Accounting.
- Avoids paperwork.
- Simple to learn and easy to use.

Clarification of important information:

The system is between the customer and waiter in the restaurant and it's not a virtual system. The waiter takes customer orders. Our system will facilitate the process of taking orders.

Application's GUI:

1. Logged in as an Admin: (Username: admin, Password: admin)

Update Prices choice:

Menu choice to see the previous updates:

2. Logged in as a Cashier: (Any username & password)

Menu choice:

Save Receipt:

All Receipt that has been saved, will be shown in the JavaApplication2 UPDATED file:

About us choice:

Database Schema:

Classes UML:

1.0.1 Configuración de la BBDD (MySQL / MariaDB)

Antes de ejecutar la aplicación, se debe tener instalado MySQL ó MariaDB y haber creado la base de datos correspondiente.

1. Asegurarse de que el gestor de base de datos está instalado y ejecutándose:
   ```
   sudo systemctl status mysql/mariadb
   sudo systemctl start mysql/mariadb
   sudo systemctl enable mysql/mariadb
   ```
2. Iniciar sesión en el gestor de base de datos:
   ```
   sudo mysql -u root -p
   ```
3. Crear la base de datos del sistema:
   ```
   CREATE DATABASE project3;
   USE project3;
   ```
4. Importar el script de creación de tablas y datos iniciales (archivo database.sql incluido en el proyecto):
   ```
   SOURCE ruta/al/proyecto/database.sql;
   ```
5. Verificar que las tablas se hayan creado correctamente:
   ```
   SHOW TABLES;
   ```
6. Ejecutar el proyecto:
   ```
   mvn exec:java
   ```

Chapter 2

Directory Hierarchy

2.1 Directories

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

File Index

6.1 File List

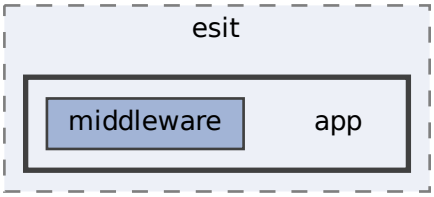
Here is a list of all files with brief descriptions:

Chapter 7

Directory Documentation

7.1 app Directory Reference

Directory dependency graph for app:



Directories

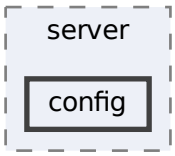
- directory middleware

Files

- file AboutUs.java
- file AdminLogin.java
- file AdminProducts.java
- file ApplicationLauncher.java
- file CashierLogin.java
- file Login.java
- file Order.java

7.2 config Directory Reference

Directory dependency graph for config:

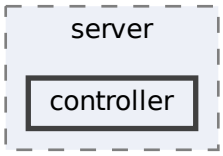


Files

- file SecurityConfig.java

7.3 controller Directory Reference

Directory dependency graph for controller:



Files

- file AppetizerController.java
- file AuthController.java
- file CashierController.java
- file DrinkController.java
- file HealthController.java
- file MainCourseController.java
- file MenuController.java

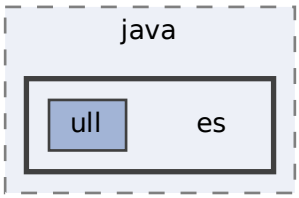
7.4 db Directory Reference

Files

- file 00-create-db.sql
- file 01-tables.sql
- file 02-procedures.sql
- file 03-triggers.sql
- file 04-privileges.sql
- file 05-data.sql
- file init.sql

7.5 es Directory Reference

Directory dependency graph for es:

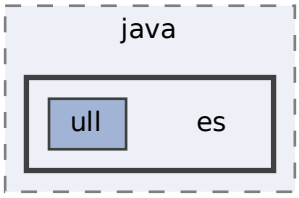


Directories

- directory ull

7.6 es Directory Reference

Directory dependency graph for es:

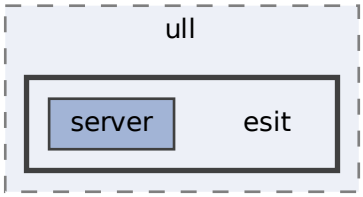


Directories

- directory ull

7.7 esit Directory Reference

Directory dependency graph for esit:

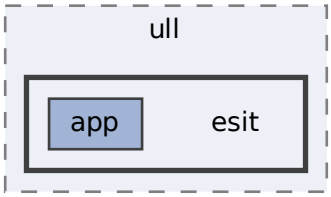


Directories

- directory server

7.8 esit Directory Reference

Directory dependency graph for esit:

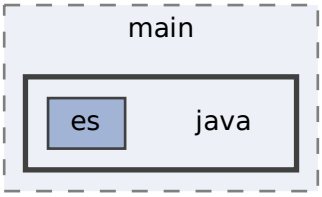


Directories

- directory app

7.9 java Directory Reference

Directory dependency graph for java:

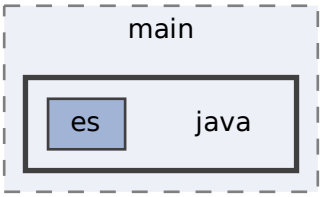


Directories

- directory es

7.10 java Directory Reference

Directory dependency graph for java:

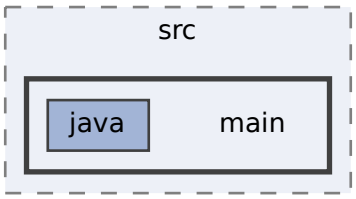


Directories

- directory es

7.11 main Directory Reference

Directory dependency graph for main:

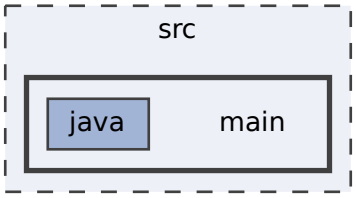


Directories

- directory java

7.12 main Directory Reference

Directory dependency graph for main:

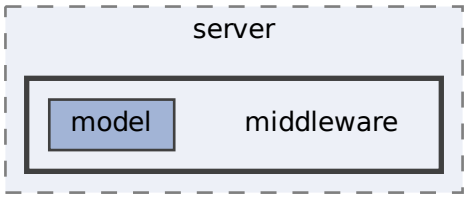


Directories

- directory java

7.13 middleware Directory Reference

Directory dependency graph for middleware:

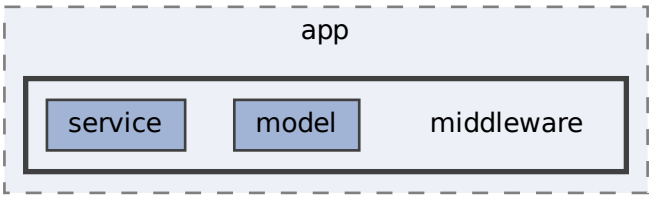


Directories

- directory model

7.14 middleware Directory Reference

Directory dependency graph for middleware:



Directories

- directory model
- directory service

Files

- file ApiClient.java
- file ApiClientException.java

7.15 model Directory Reference

Directory dependency graph for model:

Files

- file Appetizer.java
- file Cashier.java
- file Drink.java
- file MainCourse.java
- file User.java

7.16 model Directory Reference

Directory dependency graph for model:

Files

- file Appetizer.java
- file BillResult.java
- file Cashier.java
- file Drink.java
- file MainCourse.java
- file User.java

7.17 repo Directory Reference

Directory dependency graph for repo:

Files

- file AppetizerRepository.java
- file CashierRepository.java
- file DrinkRepository.java
- file MainCourseRepository.java
- file UserRepository.java

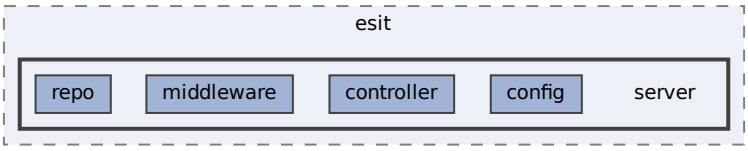
7.18 server Directory Reference

Directories

- directory src

7.19 server Directory Reference

Directory dependency graph for server:



Directories

- directory config
- directory controller
- directory middleware
- directory repo

Files

- file RestaurantApplication.java

7.20 service Directory Reference

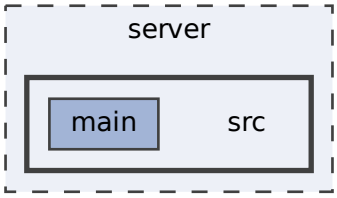
Directory dependency graph for service:

Files

- file AuthService.java
- file OrderService.java
- file ProductService.java
- file ReportService.java

7.21 src Directory Reference

Directory dependency graph for src:



Directories

- directory main

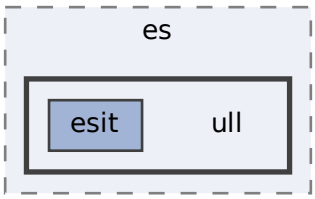
7.22 src Directory Reference

Directories

- directory main

7.23 ull Directory Reference

Directory dependency graph for ull:

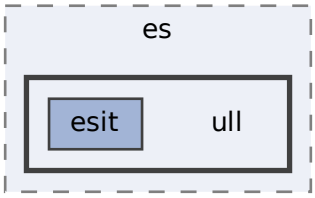


Directories

- directory esit

7.24 ull Directory Reference

Directory dependency graph for ull:



Directories

- directory esit

Chapter 8

Namespace Documentation

8.1 Package es.ull.esit.app

Packages

- package middleware

Classes

- class AboutUs

  "About us" window displaying the restaurant's history and info.
- class AdminLogin

  Login window for authenticating administrators.
- class AdminProducts

  Administrative window for managing products and prices.
- class ApplicationLauncher

  Auxiliary entry point used to start the application's UI.
- class CashierLogin

  Login window for authenticating cashiers.
- class Login

  Login window for the Restaurant System.
- class Order

  Main menu window for taking customer orders.

8.2 Package es.ull.esit.app.middleware

Packages

- package model
- package service

Classes

- class ApiClient

  API client for interacting with the backend REST API.
- class ApiClientException

  Custom exception class for API client errors.

8.3 Package es.ull.esit.app.middleware.model

Classes

- class Appetizer

  Client-side model representing an appetizer received from the backend API.
- class BillResult

  Data Transfer Object representing the result of a bill calculation.
- class Cashier

  Client-side model representing a cashier returned by the backend.
- class Drink

  Client-side model representing a drink returned by the backend API.
- class MainCourse

  Client-side model representing a main course returned by the backend.
- class User

  Client-side model representing an authenticated user.

8.4 Package es.ull.esit.app.middleware.service

Classes

- class AuthService

  Service responsible for handling authentication logic on the client side.
- class OrderService

  Service that handles order-related calculations and receipt generation.
- class ProductService

  Service handling business logic for menu products.
- class ReportService

  Service providing reporting and system status operations.

8.5 Package es.ull.esit.server

Packages

- package config
- package controller
- package repo

Classes

- class RestaurantApplication

  Main Spring Boot application class for the restaurant backend.

8.6 Package es.ull.esit.server.config

Classes

- class SecurityConfig

  Spring Security configuration for the backend.

8.7 Package es.ull.esit.server.controller

Classes

- class AppetizerController

  REST controller for managing appetizers.
- class AuthController

  Rest controller for handling authentication-related requests.
- class CashierController

  REST controller for managing cashiers.
- class DrinkController

  REST controller for managing drinks.
- class HealthController

  REST controller for health and database connectivity checks.
- class MainCourseController

  REST controller for managing main courses.
- class MenuController

  REST controller that exposes a consolidated restaurant menu.

8.8 Package es.ull.esit.server.middleware.model

Classes

- class Appetizer

  JPA entity that represents an appetizer in the menu.
- class Cashier

  JPA entity that represents a cashier in the system.
- class Drink

  JPA entity that represents a drink in the menu.
- class MainCourse

  JPA entity that represents a main course in the menu.
- class User

  JPA entity that represents a user in the system.

8.9 Package es.ull.esit.server.repo

Classes

- interface AppetizerRepository

  Repository interface for managing appetizers in the database.
- interface CashierRepository

  Repository interface for managing cashiers in the database.
- interface DrinkRepository

  Repository interface for managing drinks in the database.
- interface MainCourseRepository

  Repository interface for managing main courses in the database.
- interface UserRepository

  Repository interface for accessing users in the database.

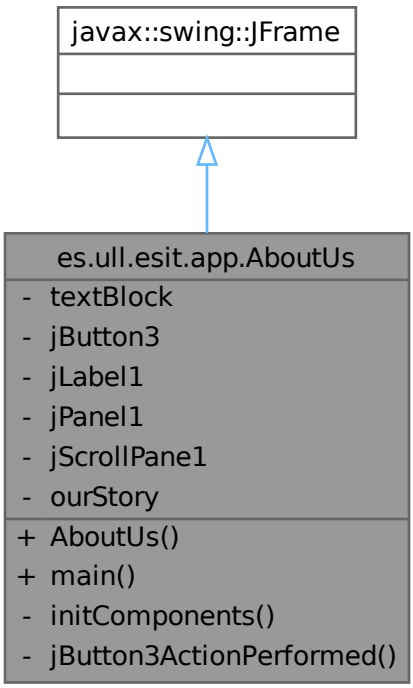
Chapter 9

Class Documentation

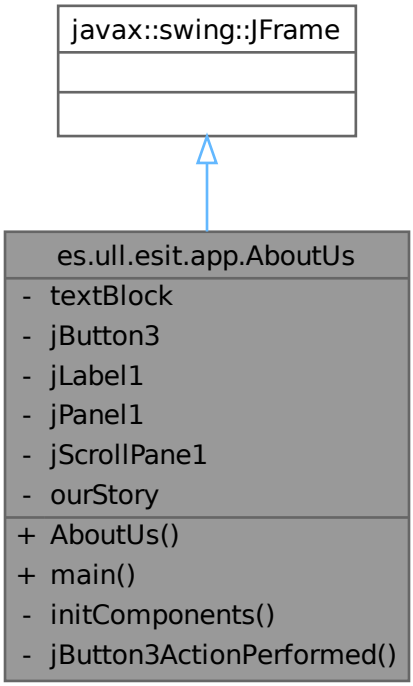
9.1 es.ull.esit.app.AboutUs Class Reference

"About us" window displaying the restaurant's history and info.

Inheritance diagram for es.ull.esit.app.AboutUs:



Collaboration diagram for es.ull.esit.app.AboutUs:



Public Member Functions

- AboutUs ()

  Default constructor.

Static Public Member Functions

- static void main (String[ ] args)

  Standalone entry point for testing the Info window.

Private Member Functions

- void initComponents ()

  Initializes and lays out Swing components.
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)

  Handler for the "Go Back" button.

Private Attributes

- String textBlock

  Static HTML content for the "Our Story" section.
- javax.swing.JButton jButton3

  Button that navigates back to the cashier login window.
- javax.swing.JLabel jLabel1

  Logo or image displayed at the top of the Info window.
- javax.swing.JPanel jPanel1

  Main container panel for all components in the Info window.
- javax.swing.JScrollPane jScrollPane1

  Scroll pane that wraps the static "ourStory" text area.
- javax.swing.JTextArea ourStory

  Text area containing the restaurant history and description.

9.1.1 Detailed Description

"About us" window displaying the restaurant's history and info.

```
Simple Swing window that:
- shows a static text area with the project story ("Our Story").
- includes a navigation button to return to the previous screen
CashierLogin.

Does not perform any network or database operations:
all the content is embedded directly in the text area.
```

Definition at line 14 of file AboutUs.java.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 AboutUs()

```
es.ull.esit.app.AboutUs.AboutUs ()  [inline]
```

Default constructor.

```
Creates an instance of the Info window and UI components.
```

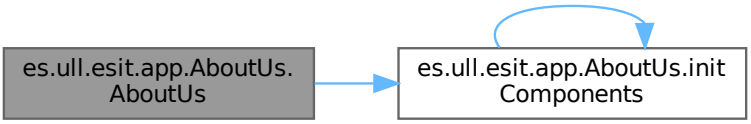
Definition at line 35 of file AboutUs.java.

```
00035                     {
00036     initComponents();
00037   }
```

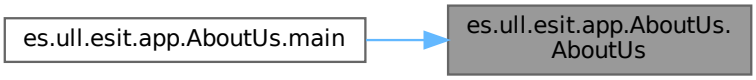
References initComponents().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.3 Member Function Documentation

9.1.3.1 initComponents()

```
void es.ull.esit.app.AboutUs.initComponents ()  [inline], [private]
```

Initializes and lays out Swing components.

```
   Generates by the Form Editor: don't modify manually.

   Sets up the JFrame, including the main panel, the static text area
   with the
   restaurant story, the logo/image label, and the "Go Back" button.
   Also configure the layout and event handlers.
```

Definition at line 53 of file AboutUs.java.

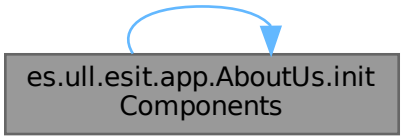
```
00053                                {
00054
00055     jPanel1 = new javax.swing.JPanel();
00056     jButton3 = new javax.swing.JButton();
00057     jScrollPane1 = new javax.swing.JScrollPane();
00058     ourStory = new javax.swing.JTextArea();
00059     jLabel1 = new javax.swing.JLabel();
00060
00061     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00062     setTitle("Info");
00063     setResizable(false);
00064
00065     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00066
00067     jButton3.setBackground(new java.awt.Color(255, 255, 255));
00068     jButton3.setFont(new java.awt.Font("Yu Gothic UI", 1, 18)); // NOI18N
00069     jButton3.setText("Go Back");
00070     jButton3.addActionListener(this::jButton3ActionPerformed);
```

```
00071
00072     ourStory.setEditable(false);
00073     ourStory.setBackground(new java.awt.Color(248, 244, 230));
00074     ourStory.setColumns(20);
00075     ourStory.setFont(new java.awt.Font("Yu Gothic UI Light", 0, 14)); // NOI18N
00076     ourStory.setRows(5);
00077     ourStory.setText(textBlock);
00078     ourStory.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
00079     jScrollPane1.setViewportView(ourStory);
00080
00081     // Updated path to match other UI resources
00082     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png")));
00083
00084     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00085     jPanel1.setLayout(jPanel1Layout);
00086     jPanel1Layout.setHorizontalGroup(
00087         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00088             .addGroup(jPanel1Layout.createSequentialGroup()
00089                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00090                     .addGroup(jPanel1Layout.createSequentialGroup()
00091                         .addGap(55, 55, 55)
00092
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00093                             .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00094                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00095                             .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 1004,
00096                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00097                     .addGroup(jPanel1Layout.createSequentialGroup()
00098                         .addGap(489, 489, 489)
00099                         .addComponent(jLabel1)))
00100                 .addContainerGap(159, Short.MAX_VALUE)));
00101     jPanel1Layout.setVerticalGroup(
00102         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00103             .addGroup(jPanel1Layout.createSequentialGroup()
00104                 .addContainerGap(71, Short.MAX_VALUE)
00105                 .addComponent(jLabel1)
00106                 .addGap(67, 67, 67)
00107                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00108                     javax.swing.GroupLayout.PREFERRED_SIZE)
00109                 .addGap(26, 26, 26)
00110                 .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00111                     javax.swing.GroupLayout.PREFERRED_SIZE)
00112                 .addGap(30, 30, 30)));
00113
00114     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00115     getContentPane().setLayout(layout);
00116     layout.setHorizontalGroup(
00117         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00118             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
00119                 Short.MAX_VALUE));
00120     layout.setVerticalGroup(
00121         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00122             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
00123                 Short.MAX_VALUE));
00124
00125     pack();
00126     setLocationRelativeTo(null);
00127   }// </editor-fold>//GEN-END:initComponents
```

References initComponents(), jButton3, jLabel1, jPanel1, jScrollPane1, ourStory, and textBlock.

Referenced by AboutUs(), and initComponents().

Here is the call graph for this function:



Here is the caller graph for this function:

9.1.3.2 jButton3ActionPerformed()

```
void es.ull.esit.app.AboutUs.jButton3ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Handler for the "Go Back" button.

```
Closes the current Info window and opens the CashierLogin
window, returning the user to the cashier login screen.
```

Parameters

evt	Action event triggered by button click.

Definition at line 137 of file AboutUs.java.

```
00137                                                                {//
      GEN-FIRST:event_jButton3ActionPerformed
00138     new CashierLogin().setVisible(true);
00139     this.dispose(); // Close current window
00140   }// GEN-LAST:event_jButton3ActionPerformed
```

9.1.3.3 main()

```
void es.ull.esit.app.AboutUs.main (
            String[] args)  [inline], [static]
```

Standalone entry point for testing the Info window.

```
Sets the Nimbus look and feel if available and shows an
instance of Info. In the normal application flow this
window is opened from CashierLogin via the "About us"
button.
```

Parameters

args	Command line arguments (not used).

Definition at line 152 of file AboutUs.java.

```
00152                                    {
00153     try {
```

```
00154       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00155         if ("Nimbus".equals(info.getName())) {
00156           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00157           break;
00158         }
00159       }
00160     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00161         | javax.swing.UnsupportedLookAndFeelException ex) {
00162       java.util.logging.Logger.getLogger(AboutUs.class.getName())
00163           .log(java.util.logging.Level.SEVERE, null, ex);
00164     }
00165     // </editor-fold>
00166
00167     /* Create and display the form */
00168     java.awt.EventQueue.invokeLater(() -> new AboutUs().setVisible(true));
00169   }
```

References AboutUs().

Here is the call graph for this function:

9.1.4 Member Data Documentation

9.1.4.1 jButton3

```
javax.swing.JButton es.ull.esit.app.AboutUs.jButton3  [private]
```

Button that navigates back to the cashier login window.

Definition at line 175 of file AboutUs.java.

Referenced by initComponents().

9.1.4.2 jLabel1

```
javax.swing.JLabel es.ull.esit.app.AboutUs.jLabel1  [private]
```

Logo or image displayed at the top of the Info window.

Definition at line 177 of file AboutUs.java.

Referenced by initComponents().

9.1.4.3 jPanel1

```
javax.swing.JPanel es.ull.esit.app.AboutUs.jPanel1  [private]
```

Main container panel for all components in the Info window.

Definition at line 179 of file AboutUs.java.

Referenced by initComponents().

9.1.4.4 jScrollPane1

```
javax.swing.JScrollPane es.ull.esit.app.AboutUs.jScrollPane1  [private]
```

Scroll pane that wraps the static "ourStory" text area.

Definition at line 181 of file AboutUs.java.

Referenced by initComponents().

9.1.4.5 ourStory

```
javax.swing.JTextArea es.ull.esit.app.AboutUs.ourStory  [private]
```

Text area containing the restaurant history and description.

Definition at line 183 of file AboutUs.java.

Referenced by initComponents().

9.1.4.6 textBlock

```
String es.ull.esit.app.AboutUs.textBlock  [private]
```

Initial value:

```
= """
      <html>
          <body>
              <tag>
                  <h1>Our Story</h1>
                  <p>Collecting flavors from all over the world to give everyone a taste of what they
      love.</p>
                  <p>Black Plate came in with this vision in mind, to be more of a home rather than an
      establishment.</p>
                  <p>2021 marked the beginning of the journey of Black Plate, starting from Al Khobar.</p>
                  <p>Because your visit means a lot to us, we would love to hear your suggestions and
      concerns so we can improve!</p>
              </tag>
          </body>
      </html>"""
```

Static HTML content for the "Our Story" section.

Definition at line 17 of file AboutUs.java.

Referenced by initComponents().

The documentation for this class was generated from the following file:

- AboutUs.java

9.2 es.ull.esit.app.AdminLogin Class Reference

Login window for authenticating administrators.

Inheritance diagram for es.ull.esit.app.AdminLogin:

Collaboration diagram for es.ull.esit.app.AdminLogin:

Public Member Functions

- AdminLogin ()

  Constructor.

Static Public Member Functions

- static void main (String[ ] args)

  Standalone entry point for testing the AdminLogin window.

Private Member Functions

- void initComponents ()

Initializes GUI components.

- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the "Update Prices" button.
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the "Menu" button.
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the "LogOut" button.

Private Attributes

- javax.swing.JButton jButton1

  Button that opens the main menu window.
- javax.swing.JButton jButton2

  Button that opens the price management window.
- javax.swing.JButton jButton3

  Button used to log out and return to the login screen.
- javax.swing.JLabel jLabel1

  Label that displays the application logo or image.
- javax.swing.JLabel jLabel3

  Label that displays the welcome text for the administrator.
- javax.swing.JPanel jPanel1

  Main container panel for all UI elements.

Static Private Attributes

- static final String PRIMARY_FONT = "Yu Gothic UI"

  Primary font used in the UI components.
- static final Logger LOGGER = LoggerFactory.getLogger(AdminLogin.class)

  Logger for the class.

9.2.1 Detailed Description

Login window for authenticating administrators.

```
It is displayed after a successful login with role ADMIN.
Shows a Swing form that lets administrators:
- access to the product and price management window.
- access to the general menu window used to place orders.
- log out and return to the login window.
```

Definition at line 16 of file AdminLogin.java.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 AdminLogin()

```
es.ull.esit.app.AdminLogin.AdminLogin ()  [inline]
```

Constructor.

```
   Creates the admin login window and initializes GUI components.
```

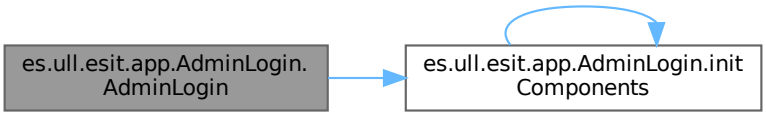
Definition at line 38 of file AdminLogin.java.

```
00038                       {
00039     initComponents();
00040   }
```

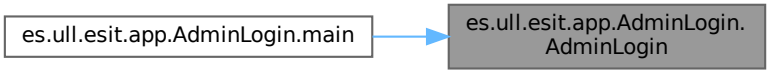
References initComponents().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.2.3 Member Function Documentation

9.2.3.1 initComponents()

```
void es.ull.esit.app.AdminLogin.initComponents ()  [inline], [private]
```

Initializes GUI components.

Generated by the Form Editor: do not modify.
Sets up welcome label, logo image, buttons for "Update Prices",
"Menu", and "LogOut" and the layout and configuration of the window.

Definition at line 51 of file AdminLogin.java.

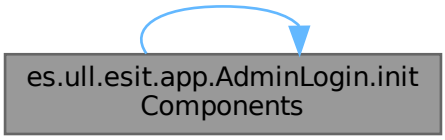
```
00051                               {
00052
00053     jPanel1 = new javax.swing.JPanel();
00054     jLabel3 = new javax.swing.JLabel();
00055     jLabel1 = new javax.swing.JLabel();
00056     jButton2 = new javax.swing.JButton();
00057     jButton1 = new javax.swing.JButton();
00058     jButton3 = new javax.swing.JButton();
00059
00060     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00061     setTitle("Admin ");
00062     setResizable(false);
00063
00064     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00065
00066     jLabel3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 24)); // NOI18N
00067     jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00068     jLabel3.setText("Welcome Admin");
00069
00070     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
      NOI18N
00071
00072     jButton2.setBackground(new java.awt.Color(153, 153, 153));
00073     jButton2.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00074     jButton2.setText("Update Prices");
00075     jButton2.addActionListener(this::jButton2ActionPerformed);
00076
00077     jButton1.setBackground(new java.awt.Color(153, 153, 153));
00078     jButton1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00079     jButton1.setText("Menu");
00080     jButton1.addActionListener(this::jButton1ActionPerformed);
00081
00082     jButton3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00083     jButton3.setText("LogOut");
00084     jButton3.addActionListener(this::jButton3ActionPerformed);
00085
00086     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00087     jPanel1.setLayout(jPanel1Layout);
00088     jPanel1Layout.setHorizontalGroup(
00089         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00090             .addGroup(jPanel1Layout.createSequentialGroup()
00091                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00092                     .addGroup(jPanel1Layout.createSequentialGroup()
00093                         .addGap(140, 140, 140)
00094                         .addComponent(jLabel1))
00095                     .addGroup(jPanel1Layout.createSequentialGroup()
00096                         .addGap(66, 66, 66)
00097
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00098                             .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00099                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00100                             .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00101                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00102                             .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 306,
00103                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00104                     .addGroup(jPanel1Layout.createSequentialGroup()
00105                         .addGap(152, 152, 152)
00106                         .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00107                             javax.swing.GroupLayout.PREFERRED_SIZE)))
00108                 .addContainerGap(69, Short.MAX_VALUE)));
00109     jPanel1Layout.setVerticalGroup(
00110         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00111             .addGroup(jPanel1Layout.createSequentialGroup()
00112                 .addGap(31, 31, 31)
00113                 .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
00114                     javax.swing.GroupLayout.PREFERRED_SIZE)
00115                 .addGap(18, 18, 18)
00116                 .addComponent(jLabel1)
00117                 .addGap(29, 29, 29)
00118                 .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00119                     javax.swing.GroupLayout.PREFERRED_SIZE)
00120                 .addGap(18, 18, 18)
00121                 .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00122                     javax.swing.GroupLayout.PREFERRED_SIZE)
00123                 .addGap(29, 29, 29)
00124                 .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00125                     javax.swing.GroupLayout.PREFERRED_SIZE)
00126                 .addContainerGap(115, Short.MAX_VALUE)));
00127
```

```
00128     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00129     getContentPane().setLayout(layout);
00130     layout.setHorizontalGroup(
00131         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00132             .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00133                 javax.swing.GroupLayout.PREFERRED_SIZE));
00134     layout.setVerticalGroup(
00135         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00136             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00137                 Short.MAX_VALUE));
00138
00139     pack();
00140     setLocationRelativeTo(null);
00141   }// </editor-fold>//GEN-END:initComponents
```

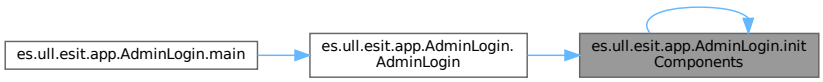
References initComponents(), jButton1, jButton2, jButton3, jLabel1, jLabel3, jPanel1, and PRIMARY_FONT.

Referenced by AdminLogin(), and initComponents().

Here is the call graph for this function:



Here is the caller graph for this function:



9.2.3.2 jButton1ActionPerformed()

```
void es.ull.esit.app.AdminLogin.jButton1ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Menu" button.

Actions steps:

- Opens the general menu window to place orders.

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by button click.

Definition at line 176 of file AdminLogin.java.

```
00176                                                                    {//
      GEN-FIRST:event_jButton1ActionPerformed
00177   try {
00178     orderSupplier.get().setVisible(true);
00179     this.dispose();
00180   } catch (Exception ex) {
00181         LOGGER.error("Error opening menu window", ex);
00182         javax.swing.JOptionPane.showMessageDialog(
00183             this,
00184             "Error opening menu window:\n" + ex.getMessage(),
00185             "Error",
00186             javax.swing.JOptionPane.ERROR_MESSAGE
00187         );
00188     }
00189   }// GEN-LAST:event_jButton1ActionPerformed
```

References LOGGER.

9.2.3.3 jButton2ActionPerformed()

```
void es.ull.esit.app.AdminLogin.jButton2ActionPerformed (
             java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Update Prices" button.

```
Actions steps:
- Opens the product administration window to modify prices and
products.
```

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by button click.

Definition at line 153 of file AdminLogin.java.

```
00153                                                                    {//
      GEN-FIRST:event_jButton2ActionPerformed
00154   try {
00155     adminProductsSupplier.get().setVisible(true);
00156     this.dispose();
00157   } catch (Exception ex) {
00158         LOGGER.error("Error opening product admin window", ex);
00159         javax.swing.JOptionPane.showMessageDialog(
00160             this,
00161             "Error opening product admin window:\n" + ex.getMessage(),
00162             "Error",
00163             javax.swing.JOptionPane.ERROR_MESSAGE
00164         );
00165     }
00166   }// GEN-LAST:event_jButton2ActionPerformed
```

References LOGGER.

9.2.3.4 jButton3ActionPerformed()

```
void es.ull.esit.app.AdminLogin.jButton3ActionPerformed (
             java.awt.event.ActionEvent evt)  [inline], [private]
```


Action handler for the "LogOut" button.

Actions steps:

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by button click.

Definition at line 199 of file AdminLogin.java.

```
00199                                                                    {//
      GEN-FIRST:event_jButton3ActionPerformed
00200   loginSupplier.get().setVisible(true);
00201   this.dispose();
00202   }// GEN-LAST:event_jButton3ActionPerformed
```

9.2.3.5 main()

```
void es.ull.esit.app.AdminLogin.main (
             String[] args)  [inline], [static]
```

Standalone entry point for testing the AdminLogin window.

```
   Sets the Nimbus look and feel if available and shows an instance
   of AdminLogin. In the normal application flow this window is started
   from the Login class after a successful admin login.
```

Parameters

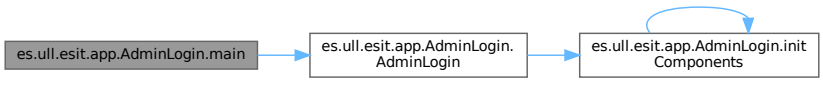
args	[String[]] Command line arguments (not used).

Definition at line 213 of file AdminLogin.java.

```
00213                                       {
00214     try {
00215       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00216         if ("Nimbus".equals(info.getName())) {
00217           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00218           break;
00219         }
00220       }
00221     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00222         | javax.swing.UnsupportedLookAndFeelException ex) {
00223
      java.util.logging.Logger.getLogger(AdminLogin.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00224     }
00225     // </editor-fold>
00226
00227     java.awt.EventQueue.invokeLater(() -> new AdminLogin().setVisible(true));
00228   }
```

References AdminLogin().

Here is the call graph for this function:



9.2.4 Member Data Documentation

9.2.4.1 jButton1

```
javax.swing.JButton es.ull.esit.app.AdminLogin.jButton1  [private]
```

Button that opens the main menu window.

Definition at line 234 of file AdminLogin.java.

Referenced by initComponents().

9.2.4.2 jButton2

```
javax.swing.JButton es.ull.esit.app.AdminLogin.jButton2  [private]
```

Button that opens the price management window.

Definition at line 236 of file AdminLogin.java.

Referenced by initComponents().

9.2.4.3 jButton3

```
javax.swing.JButton es.ull.esit.app.AdminLogin.jButton3  [private]
```

Button used to log out and return to the login screen.

Definition at line 238 of file AdminLogin.java.

Referenced by initComponents().

9.2.4.4 jLabel1

```
javax.swing.JLabel es.ull.esit.app.AdminLogin.jLabel1  [private]
```

Label that displays the application logo or image.

Definition at line 240 of file AdminLogin.java.

Referenced by initComponents().

9.2.4.5 jLabel3

```
javax.swing.JLabel es.ull.esit.app.AdminLogin.jLabel3  [private]
```

Label that displays the welcome text for the administrator.

Definition at line 242 of file AdminLogin.java.

Referenced by initComponents().

9.2.4.6 jPanel1

```
javax.swing.JPanel es.ull.esit.app.AdminLogin.jPanel1  [private]
```

Main container panel for all UI elements.

Definition at line 244 of file AdminLogin.java.

Referenced by initComponents().

9.2.4.7 LOGGER

```
final Logger es.ull.esit.app.AdminLogin.LOGGER = LoggerFactory.getLogger(AdminLogin.class)
[static], [private]
```

Logger for the class.

Replaces use of printStackTrace for production-safe logging.

Definition at line 22 of file AdminLogin.java.

Referenced by jButton1ActionPerformed(), and jButton2ActionPerformed().

9.2.4.8 PRIMARY_FONT

```
final String es.ull.esit.app.AdminLogin.PRIMARY_FONT = "Yu Gothic UI"  [static], [private]
```

Primary font used in the UI components.

Definition at line 19 of file AdminLogin.java.

Referenced by initComponents().

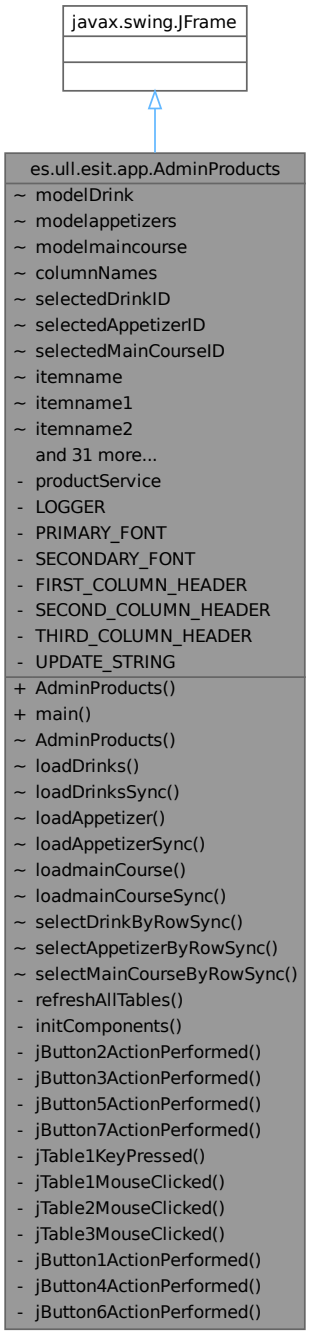
The documentation for this class was generated from the following file:

- AdminLogin.java

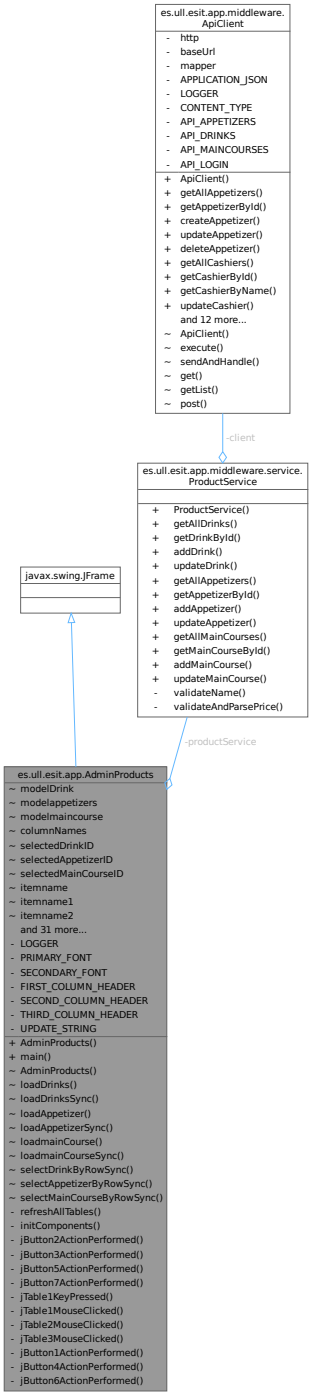
9.3 es.ull.esit.app.AdminProducts Class Reference

Administrative window for managing products and prices.

Inheritance diagram for es.ull.esit.app.AdminProducts:



Collaboration diagram for es.ull.esit.app.AdminProducts:



Public Member Functions

- AdminProducts ()

  Constructor.

Static Public Member Functions

- static void main (String[ ] args)

Standalone entry point for testing the AdminProducts window.

Private Member Functions

- void refreshAllTables ()
  Reloads all product tables.
- void initComponents ()
  Initializes GUI components.
- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
  Action handler for the "Add" button in the Drinks tab.
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)
  Action handler for the "Go Back" button.
- void jButton5ActionPerformed (java.awt.event.ActionEvent evt)
  Action handler for the "Add" button in the Appetizers tab.
- void jButton7ActionPerformed (java.awt.event.ActionEvent evt)
  Action handler for the "Add" button in the MainCourse tab.
- void jTable1KeyPressed (java.awt.event.KeyEvent evt)
  Key handler for the drinks table.
- void jTable1MouseClicked (java.awt.event.MouseEvent evt)
  Mouse handler for the drinks table.
- void jTable2MouseClicked (java.awt.event.MouseEvent evt)
  Mouse handler for the appetizers table.
- void jTable3MouseClicked (java.awt.event.MouseEvent evt)
  Mouse handler for the main course table.
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
  Action handler for the "Update" button in the Drinks tab.
- void jButton4ActionPerformed (java.awt.event.ActionEvent evt)
  Action handler for the "Update" button in the Appetizers tab.
- void jButton6ActionPerformed (java.awt.event.ActionEvent evt)
  Action handler for the "Update" button in the MainCourse tab.

Private Attributes

- final transient ProductService productService
  Service used to call the REST API and apply product-related logic.

Static Private Attributes

- static final Logger LOGGER = LoggerFactory.getLogger(AdminProducts.class)
  Logger for logging events and errors.
- static final String PRIMARY_FONT = "Yu Gothic UI"
  Primary font used throughout the GUI.
- static final String SECONDARY_FONT = "Thonburi"
  Secondary font used for buttons and smaller text.
- static final String FIRST_COLUMN_HEADER = "ID"
  Column header for the ID column in product tables.
- static final String SECOND_COLUMN_HEADER = "Item Name"
  Column header for the name column in product tables.
- static final String THIRD_COLUMN_HEADER = "Item Price"
  Column header for the price column in product tables.
- static final String UPDATE_STRING = "Update"
  String constant for the "Update" button label.

9.3.1 Detailed Description

Administrative window for managing products and prices.

```
Swing window that allows administrators to:
- view drinks, appetizers and main courses loaded from the backend.
- add new items to each category.
- update the price and name of existing items.

All data is obtained and persisted through the ProductService, which
internally uses ApiClient to call the REST API.
```

Definition at line 27 of file AdminProducts.java.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 AdminProducts()

```
es.ull.esit.app.AdminProducts.AdminProducts ()  [inline]
```

Constructor.

```
Creates the admin products window, initializes GUI components,
configures the table models and loads the initial data from the
backend service.
```

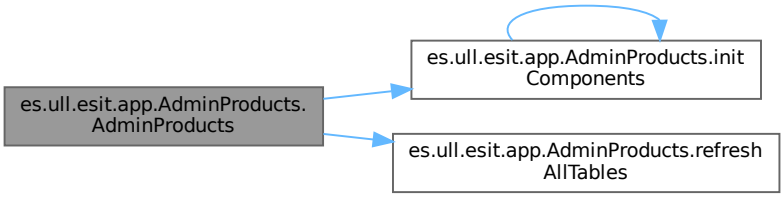
Definition at line 74 of file AdminProducts.java.

```
00074                         {
00075     initComponents();
00076
00077     // Initialize the Service Layer.
00078     ApiClient client = new ApiClient("http://localhost:8080");
00079     this.productService = new ProductService(client);
00080
00081     // Initialize table models and set shared headers.
00082     modelDrink = new DefaultTableModel();
00083     modelDrink.setColumnIdentifiers(columnNames);
00084     modelappetizers = new DefaultTableModel();
00085     modelappetizers.setColumnIdentifiers(columnNames);
00086     modelmaincourse = new DefaultTableModel();
00087     modelmaincourse.setColumnIdentifiers(columnNames);
00088
00089     // Link models to tables.
00090     jTable1.setModel(modelDrink);
00091     jTable2.setModel(modelappetizers);
00092     jTable3.setModel(modelmaincourse);
00093
00094     // Load initial data from backend.
00095     refreshAllTables();
00096   }
```

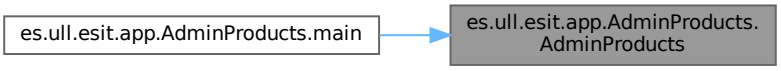
References initComponents(), and refreshAllTables().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.3 Member Function Documentation

9.3.3.1 initComponents()

```
void es.ull.esit.app.AdminProducts.initComponents ()  [inline], [private]
```

Initializes GUI components.

```
Generated by the Form Editor: do not modify manually.
Creates labels, buttons, tabbed panes and tables for the three
product categories (drinks, appetizers, main courses) and the
navigation button "Go Back".
```

Definition at line 317 of file AdminProducts.java.

```
00317                                {
00318
00319     jPanel1 = new javax.swing.JPanel();
00320     jLabel1 = new javax.swing.JLabel();
00321     jTabbedPane1 = new javax.swing.JTabbedPane();
00322     jPanel2 = new javax.swing.JPanel();
00323     itemname = new javax.swing.JTextField();
00324     itemprice = new javax.swing.JTextField();
00325     jScrollPane1 = new javax.swing.JScrollPane();
00326     jTable1 = new javax.swing.JTable();
00327     jButton1 = new javax.swing.JButton();
00328     jButton2 = new javax.swing.JButton();
00329     jLabel2 = new javax.swing.JLabel();
00330     jLabel3 = new javax.swing.JLabel();
00331     jLabel4 = new javax.swing.JLabel();
00332     jPanel3 = new javax.swing.JPanel();
00333     jLabel5 = new javax.swing.JLabel();
00334     jLabel6 = new javax.swing.JLabel();
00335     jLabel7 = new javax.swing.JLabel();
00336     jScrollPane2 = new javax.swing.JScrollPane();
```

```
00337     jTable2 = new javax.swing.JTable();
00338     itemname1 = new javax.swing.JTextField();
00339     itemprice1 = new javax.swing.JTextField();
00340     jButton4 = new javax.swing.JButton();
00341     jButton5 = new javax.swing.JButton();
00342     jPanel4 = new javax.swing.JPanel();
00343     jLabel8 = new javax.swing.JLabel();
00344     jLabel9 = new javax.swing.JLabel();
00345     jLabel10 = new javax.swing.JLabel();
00346     jScrollPane3 = new javax.swing.JScrollPane();
00347     jTable3 = new javax.swing.JTable();
00348     itemname2 = new javax.swing.JTextField();
00349     itemprice2 = new javax.swing.JTextField();
00350     jButton6 = new javax.swing.JButton();
00351     jButton7 = new javax.swing.JButton();
00352     jButton3 = new javax.swing.JButton();
00353 
00354     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00355     setTitle("Admin");
00356     setResizable(false);
00357 
00358     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00359 
00360     jLabel1.setBackground(new java.awt.Color(255, 153, 0));
00361     jLabel1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 36)); // NOI18N
00362     jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00363     jLabel1.setText("Items Prices Update Portal");
00364 
00365     jTabbedPane1.setBackground(new java.awt.Color(248, 244, 230));
00366     jTabbedPane1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true));
00367     jTabbedPane1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
00368     jTabbedPane1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00369 
00370     jPanel2.setBackground(new java.awt.Color(248, 244, 230));
00371 
00372     itemname.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00373     itemname.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00374     itemname.setBorder(javax.swing.BorderFactory.createTitledBorder(
00375         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), SECOND_COLUMN_HEADER,
00376         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00377         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00378 
00379     itemprice.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00380     itemprice.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00381     itemprice.setBorder(javax.swing.BorderFactory.createTitledBorder(
00382         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), THIRD_COLUMN_HEADER,
00383         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00384         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00385 
00386     jTable1.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00387     jTable1.setModel(new javax.swing.table.DefaultTableModel(
00388         new Object[][] {
00389             {},
00390             {},
00391             {},
00392             {}
00393         },
00394         new String[] {
00395 
00396         }));
00397     jTable1.setInheritsPopupMenu(true);
00398     jTable1.getTableHeader().setResizingAllowed(false);
00399     jTable1.getTableHeader().setReorderingAllowed(false);
00400     jTable1.addMouseListener(new java.awt.event.MouseAdapter() {
00401       @Override
00402       public void mouseClicked(java.awt.event.MouseEvent evt) {
00403         jTable1MouseClicked(evt);
00404       }
00405     });
00406     jTable1.addKeyListener(new java.awt.event.KeyAdapter() {
00407       @Override
00408       public void keyPressed(java.awt.event.KeyEvent evt) {
00409         jTable1KeyPressed(evt);
00410       }
00411     });
00412     jScrollPane1.setViewportView(jTable1);
00413 
00414     jButton1.setBackground(new java.awt.Color(255, 255, 255));
00415     jButton1.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00416     jButton1.setText(UPDATE_STRING);
00417     jButton1.addActionListener(this::jButton1ActionPerformed);
00418 
00419     jButton2.setBackground(new java.awt.Color(255, 255, 255));
00420     jButton2.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00421     jButton2.setText("Add");
00422     jButton2.addActionListener(this::jButton2ActionPerformed);
00423 
```

```
00424     jLabel2.setBackground(new java.awt.Color(255, 153, 0));
00425     jLabel2.setFont(new java.awt.Font(PRIMARY_FONT, 1, 25)); // NOI18N
00426     jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00427     jLabel2.setText("Available Drinks");
00428 
00429     jLabel3.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00430     jLabel3.setText("Enter new drink information carefully to add.");
00431 
00432     jLabel4.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00433     jLabel4.setText("Please select the drink to update the price.");
00434 
00435     javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
00436     jPanel2.setLayout(jPanel2Layout);
00437     jPanel2Layout.setHorizontalGroup(
00438         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00439             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel2Layout.createSequentialGroup()
00440                 .addGap(0, 27, Short.MAX_VALUE)
00441                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00442                     .addGroup(jPanel2Layout.createSequentialGroup()
00443                         .addGap(24, 24, 24)
00444 
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00445                             .addGroup(jPanel2Layout.createSequentialGroup()
00446                                 .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
00447                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00448                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
      109,
00449                                     Short.MAX_VALUE)
00450                                 .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
00451                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00452                                 .addGap(8, 8, 8))
00453                             .addComponent(itemprice)
00454                             .addComponent(itemname, javax.swing.GroupLayout.DEFAULT_SIZE, 311,
      Short.MAX_VALUE)
00455                             .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
00456                                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00457                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00458                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00459 
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00460                             .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 586,
00461                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00462                             .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 408,
00463                                 javax.swing.GroupLayout.PREFERRED_SIZE))
00464                         .addGap(115, 115, 115))
00465                     .addGroup(jPanel2Layout.createSequentialGroup()
00466                         .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 1111,
00467                             javax.swing.GroupLayout.PREFERRED_SIZE)
00468                         .addGap(18, 18, 18)))));
00469     jPanel2Layout.setVerticalGroup(
00470         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00471             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel2Layout.createSequentialGroup()
00472                 .addContainerGap()
00473                 .addComponent(jLabel2)
00474                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
      Short.MAX_VALUE)
00475 
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00476                     .addComponent(jLabel3)
00477                     .addComponent(jLabel4))
00478                 .addGap(18, 18, 18)
00479                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00480                     .addGroup(jPanel2Layout.createSequentialGroup()
00481                         .addComponent(itemname, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00482                             javax.swing.GroupLayout.PREFERRED_SIZE)
00483                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00484                         .addComponent(itemprice, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00485                             javax.swing.GroupLayout.PREFERRED_SIZE)
00486                         .addGap(37, 37, 37)
00487 
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00488                             .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00489                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00490                             .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00491                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00492                     .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00493                         javax.swing.GroupLayout.PREFERRED_SIZE))
00494                 .addGap(48, 48, 48)));
00495 
00496     jTabbedPane1.addTab("Drinks", jPanel2);
00497 
00498     jPanel3.setBackground(new java.awt.Color(248, 244, 230));
00499 
00500     jLabel5.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00501     jLabel5.setText("Enter new appetizer information carefully to add.");
```

```
00502
00503     jLabel6.setBackground(new java.awt.Color(255, 153, 0));
00504     jLabel6.setFont(new java.awt.Font(PRIMARY_FONT, 1, 25)); // NOI18N
00505     jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00506     jLabel6.setText("Available Appetizer");
00507
00508     jLabel7.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00509     jLabel7.setText("Please select the appetizer to update the price.");
00510
00511     jTable2.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00512     jTable2.setModel(new javax.swing.table.DefaultTableModel(
00513         new Object[][] {
00514             {},
00515             {},
00516             {},
00517             {}
00518         },
00519         new String[] {
00520
00521         }));
00522     jTable2.addMouseListener(new java.awt.event.MouseAdapter() {
00523       @Override
00524       public void mouseClicked(java.awt.event.MouseEvent evt) {
00525         jTable2MouseClicked(evt);
00526       }
00527     });
00528     jScrollPane2.setViewportView(jTable2);
00529
00530     itemname1.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00531     itemname1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00532     itemname1.setBorder(javax.swing.BorderFactory.createTitledBorder(
00533         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), SECOND_COLUMN_HEADER,
00534         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00535         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00536
00537     itemprice1.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00538     itemprice1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00539     itemprice1.setBorder(javax.swing.BorderFactory.createTitledBorder(
00540         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), THIRD_COLUMN_HEADER,
00541         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00542         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00543
00544     jButton4.setBackground(new java.awt.Color(255, 255, 255));
00545     jButton4.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00546     jButton4.setText(UPDATE_STRING);
00547     jButton4.addActionListener(this::jButton4ActionPerformed);
00548
00549     jButton5.setBackground(new java.awt.Color(255, 255, 255));
00550     jButton5.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00551     jButton5.setText("Add");
00552     jButton5.addActionListener(this::jButton5ActionPerformed);
00553
00554     javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
00555     jPanel3.setLayout(jPanel3Layout);
00556     jPanel3Layout.setHorizontalGroup(
00557         jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00558             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel3Layout.createSequentialGroup()
00559                 .addGap(0, 27, Short.MAX_VALUE)
00560                 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00561                     .addGroup(jPanel3Layout.createSequentialGroup()
00562                         .addGap(24, 24, 24)
00563
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00564                             .addGroup(jPanel3Layout.createSequentialGroup()
00565                                 .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
00566                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00567                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
00568                                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00569                                 .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
00570                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00571                                 .addGap(8, 8, 8))
00572                             .addComponent(itemprice1)
00573                             .addComponent(itemname1)
00574                             .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE,
00575                                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00576                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00577                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00578
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00579                             .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 586,
00580                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00581                             .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 408,
00582                                 javax.swing.GroupLayout.PREFERRED_SIZE))
00583                         .addGap(115, 115, 115))
00584                     .addGroup(jPanel3Layout.createSequentialGroup()
00585                         .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 1111,
```

```
00586                           javax.swing.GroupLayout.PREFERRED_SIZE)
00587                       .addGap(18, 18, 18)))));
00588     jPanel3Layout.setVerticalGroup(
00589         jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00590             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel3Layout.createSequentialGroup()
00591                 .addContainerGap()
00592                 .addComponent(jLabel6)
00593                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
      Short.MAX_VALUE)
00594
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00595                     .addComponent(jLabel5)
00596                     .addComponent(jLabel7))
00597                 .addGap(18, 18, 18)
00598                 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00599                     .addGroup(jPanel3Layout.createSequentialGroup()
00600                         .addComponent(itemname1, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00601                             javax.swing.GroupLayout.PREFERRED_SIZE)
00602                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00603                         .addComponent(itemprice1, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00604                             javax.swing.GroupLayout.PREFERRED_SIZE)
00605                         .addGap(37, 37, 37)
00606
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00607                             .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00608                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00609                             .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00610                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00611                     .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00612                         javax.swing.GroupLayout.PREFERRED_SIZE))
00613                 .addGap(48, 48, 48)));
00614
00615     jTabbedPane1.addTab("Appetizers", jPanel3);
00616
00617     jPanel4.setBackground(new java.awt.Color(248, 244, 230));
00618     jPanel4.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00619
00620     jLabel8.setBackground(new java.awt.Color(248, 244, 230));
00621     jLabel8.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00622     jLabel8.setText("Enter new item information carefully to add.");
00623
00624     jLabel9.setBackground(new java.awt.Color(255, 153, 0));
00625     jLabel9.setFont(new java.awt.Font(PRIMARY_FONT, 1, 25)); // NOI18N
00626     jLabel9.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00627     jLabel9.setText("Avaliable MainCourse");
00628
00629     jLabel10.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00630     jLabel10.setText("Please select the item to update the price.");
00631
00632     jTable3.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00633     jTable3.setModel(new javax.swing.table.DefaultTableModel(
00634         new Object[][] {
00635             {},
00636             {},
00637             {},
00638             {}
00639         },
00640         new String[] {
00641
00642         }));
00643     jTable3.addMouseListener(new java.awt.event.MouseAdapter() {
00644       @Override
00645       public void mouseClicked(java.awt.event.MouseEvent evt) {
00646         jTable3MouseClicked(evt);
00647       }
00648     });
00649     jScrollPane3.setViewportView(jTable3);
00650
00651     itemname2.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00652     itemname2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00653     itemname2.setBorder(javax.swing.BorderFactory.createTitledBorder(
00654         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), SECOND_COLUMN_HEADER,
00655         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00656         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00657
00658     itemprice2.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00659     itemprice2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00660     itemprice2.setBorder(javax.swing.BorderFactory.createTitledBorder(
00661         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), THIRD_COLUMN_HEADER,
00662         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00663         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00664
00665     jButton6.setBackground(new java.awt.Color(255, 255, 255));
00666     jButton6.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00667     jButton6.setText(UPDATE_STRING);
00668     jButton6.addActionListener(this::jButton6ActionPerformed);
```

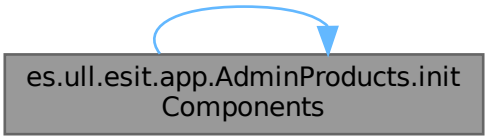
```
00669
00670     jButton7.setBackground(new java.awt.Color(255, 255, 255));
00671     jButton7.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00672     jButton7.setText("Add");
00673     jButton7.addActionListener(this::jButton7ActionPerformed);
00674
00675     javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
00676     jPanel4.setLayout(jPanel4Layout);
00677     jPanel4Layout.setHorizontalGroup(
00678         jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00679             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel4Layout.createSequentialGroup()
00680                 .addGap(0, 27, Short.MAX_VALUE)
00681                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00682                     .addGroup(jPanel4Layout.createSequentialGroup()
00683                         .addGap(24, 24, 24)
00684
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00685                             .addGroup(jPanel4Layout.createSequentialGroup()
00686                                 .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
00687                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00688                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
      109,
00689                                     Short.MAX_VALUE)
00690                                 .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
00691                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00692                                 .addGap(8, 8, 8))
00693                             .addComponent(itemprice2)
00694                             .addComponent(itemname2, javax.swing.GroupLayout.DEFAULT_SIZE, 311,
      Short.MAX_VALUE)
00695                             .addComponent(jLabel8, javax.swing.GroupLayout.DEFAULT_SIZE,
00696                                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00697                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00698                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00699
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00700                             .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 586,
00701                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00702                             .addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED_SIZE, 408,
00703                                 javax.swing.GroupLayout.PREFERRED_SIZE))
00704                         .addGap(115, 115, 115))
00705                     .addGroup(jPanel4Layout.createSequentialGroup()
00706                         .addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE, 1111,
00707                             javax.swing.GroupLayout.PREFERRED_SIZE)
00708                         .addGap(18, 18, 18)))));
00709     jPanel4Layout.setVerticalGroup(
00710         jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00711             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel4Layout.createSequentialGroup()
00712                 .addContainerGap()
00713                 .addComponent(jLabel9)
00714                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
      Short.MAX_VALUE)
00715
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00716                     .addComponent(jLabel8)
00717                     .addComponent(jLabel10))
00718                 .addGap(18, 18, 18)
00719                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00720                     .addGroup(jPanel4Layout.createSequentialGroup()
00721                         .addComponent(itemname2, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00722                             javax.swing.GroupLayout.PREFERRED_SIZE)
00723                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00724                         .addComponent(itemprice2, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00725                             javax.swing.GroupLayout.PREFERRED_SIZE)
00726                         .addGap(37, 37, 37)
00727
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00728                             .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00729                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00730                             .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00731                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00732                     .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00733                         javax.swing.GroupLayout.PREFERRED_SIZE))
00734                 .addGap(48, 48, 48)));
00735
00736     jTabbedPane1.addTab("MainCourse", jPanel4);
00737
00738     jButton3.setBackground(new java.awt.Color(255, 255, 255));
00739     jButton3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00740     jButton3.setText("Go Back");
00741     jButton3.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true));
00742     jButton3.addActionListener(this::jButton3ActionPerformed);
00743
00744     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00745     jPanel1.setLayout(jPanel1Layout);
00746     jPanel1Layout.setHorizontalGroup(
```

```
00747         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00748             .addGroup(jPanel1Layout.createSequentialGroup()
00749                 .addGap(25, 25, 25)
00750                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00751                     .addGroup(jPanel1Layout.createSequentialGroup()
00752                         .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 122,
00753                             javax.swing.GroupLayout.PREFERRED_SIZE)
00754                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
00755                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00756                         .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 689,
00757                             javax.swing.GroupLayout.PREFERRED_SIZE)
00758                         .addGap(218, 218, 218))
00759                     .addGroup(jPanel1Layout.createSequentialGroup()
00760                         .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
00761                             javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
00762                         .addContainerGap(29, Short.MAX_VALUE)))));
00763     jPanel1Layout.setVerticalGroup(
00764         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00765             .addGroup(jPanel1Layout.createSequentialGroup()
00766                 .addContainerGap()
00767
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
00768                     .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 60,
00769                         javax.swing.GroupLayout.PREFERRED_SIZE)
00770                     .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 37,
00771                         javax.swing.GroupLayout.PREFERRED_SIZE))
00772                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00773                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00774                 .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 392,
00775                     javax.swing.GroupLayout.PREFERRED_SIZE)
00776                 .addGap(29, 29, 29)));
00777
00778     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00779     getContentPane().setLayout(layout);
00780     layout.setHorizontalGroup(
00781         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00782             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00783                 Short.MAX_VALUE));
00784     layout.setVerticalGroup(
00785         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00786             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
00787                 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00788                     javax.swing.GroupLayout.PREFERRED_SIZE)
00789                 .addGap(0, 0, Short.MAX_VALUE)));
00790
00791     pack();
00792     setLocationRelativeTo(null);
00793   }// </editor-fold>//GEN-END:initComponents
```

References initComponents(), PRIMARY_FONT, SECOND_COLUMN_HEADER, and THIRD_COLUMN_HEADER.

Referenced by AdminProducts(), and initComponents().

Here is the call graph for this function:



Here is the caller graph for this function:

9.3.3.2 jButton1ActionPerformed()

```
void es.ull.esit.app.AdminProducts.jButton1ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Update" button in the Drinks tab.

```
Sends the modified drink data (name and price) to the backend,
resets the selection and reloads the drinks table.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by the button click.

Definition at line 1008 of file AdminProducts.java.

```
01008                                                                {//
      GEN-FIRST:event_jButton1ActionPerformed
01009     String name = itemname.getText();
01010     String price = itemprice.getText();
01011
01012     new Thread(() -> {
01013       try {
01014         productService.updateDrink(selectedDrinkID, name, price);
01015         SwingUtilities.invokeLater(() -> {
01016           JOptionPane.showMessageDialog(this, "Drink updated successfully.");
01017           itemname.setText("");
01018           itemprice.setText("");
01019           selectedDrinkID = null;
01020           loadDrinks();
01021         });
01022       } catch (Exception ex) {
01023         SwingUtilities
01024             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error updating drink: " +
      ex.getMessage()));
01025       }
01026     }).start();
01027   }// GEN-LAST:event_jButton1ActionPerformed
```

References productService.

9.3.3.3 jButton2ActionPerformed()

```
void es.ull.esit.app.AdminProducts.jButton2ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Add" button in the Drinks tab.

```
Steps:
- Reads the name and price from the text fields.
- Uses ProductService to create a new Drink in the backend.
- Shows a confirmation dialog and refreshes the drinks table.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by the button click.

Definition at line 805 of file AdminProducts.java.

```
00805                                                                    {//
      GEN-FIRST:event_jButton2ActionPerformed
00806     String name = itemname.getText();
00807     String price = itemprice.getText();
00808 
00809     new Thread(() -> {
00810       try {
00811         productService.addDrink(name, price);
00812         SwingUtilities.invokeLater(() -> {
00813           JOptionPane.showMessageDialog(this, "Drink Added Successfully.");
00814           itemname.setText("");
00815           itemprice.setText("");
00816           loadDrinks();
00817         });
00818       } catch (Exception ex) {
00819         SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog(this, "Error adding drink: " +
      ex.getMessage()));
00820       }
00821     }).start();
00822   }// GEN-LAST:event_jButton2ActionPerformed
```

References productService.

9.3.3.4 jButton3ActionPerformed()

```
void es.ull.esit.app.AdminProducts.jButton3ActionPerformed (
             java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Go Back" button.

```
   Closes the current AdminProducts window and returns to the
   AdminLogin window.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by the button click.

Definition at line 832 of file AdminProducts.java.

```
00832                                                                    {//
      GEN-FIRST:event_jButton3ActionPerformed
00833     new AdminLogin().setVisible(true);
00834     this.dispose();
00835   }// GEN-LAST:event_jButton3ActionPerformed
```

9.3.3.5 jButton4ActionPerformed()

```
void es.ull.esit.app.AdminProducts.jButton4ActionPerformed (
             java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Update" button in the Appetizers tab.

```
   Updates the selected appetizer in the backend and refreshes the
   appetizers table.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by the button click.

Definition at line 1037 of file AdminProducts.java.

```
01037                                                                    {//
      GEN-FIRST:event_jButton4ActionPerformed
01038     String name = itemname1.getText();
01039     String price = itemprice1.getText();
01040
01041     new Thread(() -> {
01042       try {
01043         productService.updateAppetizer(selectedAppetizerID, name, price);
01044         SwingUtilities.invokeLater(() -> {
01045           JOptionPane.showMessageDialog(this, "Appetizer updated successfully.");
01046           itemname1.setText("");
01047           itemprice1.setText("");
01048           selectedAppetizerID = null;
01049           loadAppetizer();
01050         });
01051       } catch (Exception ex) {
01052         SwingUtilities
01053             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error updating appetizer: " +
      ex.getMessage()));
01054       }
01055     }).start();
01056   }// GEN-LAST:event_jButton4ActionPerformed
```

References productService.

9.3.3.6 jButton5ActionPerformed()

```
void es.ull.esit.app.AdminProducts.jButton5ActionPerformed (
             java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Add" button in the Appetizers tab.

```
Creates a new appetizer in the backend using the data entered
by the administrator and reloads the appetizers table.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by the button click.

Definition at line 845 of file AdminProducts.java.

```
00845                                                                    {//
      GEN-FIRST:event_jButton5ActionPerformed
00846     String name = itemname1.getText();
00847     String price = itemprice1.getText();
00848
00849     new Thread(() -> {
00850       try {
00851         productService.addAppetizer(name, price);
00852         SwingUtilities.invokeLater(() -> {
00853           JOptionPane.showMessageDialog(this, "Appetizer Added Successfully.");
00854           itemname1.setText("");
00855           itemprice1.setText("");
00856           loadAppetizer();
00857         });
00858       } catch (Exception ex) {
00859         SwingUtilities
00860             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error adding appetizer: " +
      ex.getMessage()));
00861       }
00862     }).start();
00863   }// GEN-LAST:event_jButton5ActionPerformed
```

References productService.

9.3.3.7 jButton6ActionPerformed()

```
void es.ull.esit.app.AdminProducts.jButton6ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Update" button in the MainCourse tab.

```
Updates the selected main course data in the backend and reloads
the main course list.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by the button click.

Definition at line 1066 of file AdminProducts.java.

```
01066                                                                {//
      GEN-FIRST:event_jButton6ActionPerformed
01067     String name = itemname2.getText();
01068     String price = itemprice2.getText();
01069
01070     new Thread(() -> {
01071       try {
01072         productService.updateMainCourse(selectedMainCourseID, name, price);
01073         SwingUtilities.invokeLater(() -> {
01074           JOptionPane.showMessageDialog(this, "Main course updated successfully.");
01075           itemname2.setText("");
01076           itemprice2.setText("");
01077           selectedMainCourseID = null;
01078           loadmainCourse();
01079         });
01080       } catch (Exception ex) {
01081         SwingUtilities
01082             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error updating main course: " +
      ex.getMessage()));
01083       }
01084     }).start();
01085   }// GEN-LAST:event_jButton6ActionPerformed
```

References productService.

9.3.3.8 jButton7ActionPerformed()

```
void es.ull.esit.app.AdminProducts.jButton7ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Add" button in the MainCourse tab.

```
Sends a new main course to the backend and refreshes the
corresponding table.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by the button click.

Definition at line 873 of file AdminProducts.java.

```
00873                                                                {//
      GEN-FIRST:event_jButton7ActionPerformed
00874     String name = itemname2.getText();
00875     String price = itemprice2.getText();
```

```
00876
00877     new Thread(() -> {
00878       try {
00879         productService.addMainCourse(name, price);
00880         SwingUtilities.invokeLater(() -> {
00881           JOptionPane.showMessageDialog(this, "Main Course Added Successfully.");
00882           itemname2.setText("");
00883           itemprice2.setText("");
00884           loadmainCourse();
00885         });
00886       } catch (Exception ex) {
00887         SwingUtilities
00888             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error adding main course: " +
      ex.getMessage()));
00889       }
00890     }).start();
00891   }// GEN-LAST:event_jButton7ActionPerformed
```

References productService.

9.3.3.9 jTable1KeyPressed()

```
void es.ull.esit.app.AdminProducts.jTable1KeyPressed (
             java.awt.event.KeyEvent evt)  [inline], [private]
```

Key handler for the drinks table.

```
Currently not used. Kept for potential future keyboard handling
when navigating drink rows.
```

Parameters

evt	[java.awt.event.KeyEvent] Key event generated by the table.

Definition at line 901 of file AdminProducts.java.

```
00901                                                    {// GEN-FIRST:event_jTable1KeyPressed
00902     // No action needed for key press in this version.
00903   }// GEN-LAST:event_jTable1KeyPressed
```

9.3.3.10 jTable1MouseClicked()

```
void es.ull.esit.app.AdminProducts.jTable1MouseClicked (
             java.awt.event.MouseEvent evt)  [inline], [private]
```

Mouse handler for the drinks table.

```
When a row is clicked:
- Saves the selected drink ID.
- Requests detailed data from the backend.
- Fills the name and price fields so they can be edited.
```

Parameters

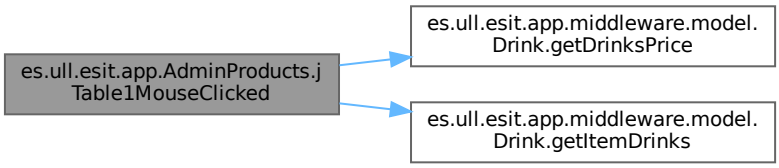
evt	[java.awt.event.MouseEvent] Mouse event generated by the table.

Definition at line 915 of file AdminProducts.java.

```
00915                                                                {//
      GEN-FIRST:event_jTable1MouseClicked
00916     int row = jTable1.getSelectedRow();
00917     if (row == -1)
00918       return;
00919
00920     String idStr = jTable1.getValueAt(row, 0).toString();
00921     selectedDrinkID = Long.valueOf(idStr);
00922     LOGGER.info("Selected Drink ID: {}", selectedDrinkID);
00923
00924     new Thread(() -> {
00925       try {
00926         Drink drink = productService.getDrinkById(selectedDrinkID);
00927         SwingUtilities.invokeLater(() -> {
00928           itemname.setText(drink.getItemDrinks());
00929           itemprice.setText(String.valueOf(drink.getDrinksPrice()));
00930         });
00931       } catch (Exception ex) {
00932         SwingUtilities
00933             .invokeLater(() -> JOptionPane.showMessageDialog(null, "Error loading drink details: " +
      ex.getMessage()));
00934       }
00935     }).start();
00936   }// GEN-LAST:event_jTable1MouseClicked
```

References es.ull.esit.app.middleware.model.Drink.getDrinksPrice(), es.ull.esit.app.middleware.model.Drink.getItemDrinks(), LOGGER, and productService.

Here is the call graph for this function:



9.3.3.11 jTable2MouseClicked()

```
void es.ull.esit.app.AdminProducts.jTable2MouseClicked (
             java.awt.event.MouseEvent evt)  [inline], [private]
```

Mouse handler for the appetizers table.

```
   Loads the selected appetizer from the backend and populates
   the corresponding text fields to allow editing.
```

Parameters

evt	[java.awt.event.MouseEvent] Mouse event generated by the table.

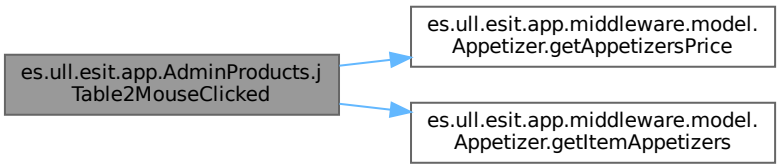
Definition at line 946 of file AdminProducts.java.

```
00946                                                                {//
      GEN-FIRST:event_jTable2MouseClicked
00947     int row = jTable2.getSelectedRow();
00948     if (row == -1)
00949       return;
```

```
00950
00951     String idStr = jTable2.getValueAt(row, 0).toString();
00952     selectedAppetizerID = Long.valueOf(idStr);
00953     LOGGER.info("Selected Appetizer ID: {}", selectedAppetizerID);
00954
00955     new Thread(() -> {
00956       try {
00957         Appetizer appetizer = productService.getAppetizerById(selectedAppetizerID);
00958         SwingUtilities.invokeLater(() -> {
00959           itemname1.setText(appetizer.getItemAppetizers());
00960           itemprice1.setText(String.valueOf(appetizer.getAppetizersPrice()));
00961         });
00962       } catch (Exception ex) {
00963         SwingUtilities.invokeLater(
00964             () -> JOptionPane.showMessageDialog(null, "Error loading appetizer details: " +
      ex.getMessage()));
00965       }
00966     }).start();
00967   }// GEN-LAST:event_jTable2MouseClicked
```

References es.ull.esit.app.middleware.model.Appetizer.getAppetizersPrice(), es.ull.esit.app.middleware.model.Appetizer.getItemAppe
LOGGER, and productService.

Here is the call graph for this function:



9.3.3.12 jTable3MouseClicked()

```
void es.ull.esit.app.AdminProducts.jTable3MouseClicked (
             java.awt.event.MouseEvent evt)  [inline], [private]
```

Mouse handler for the main course table.

```
   Loads the selected main course from the backend and puts its
   data into the editable text fields.
```

Parameters

evt	[java.awt.event.MouseEvent] Mouse event generated by the table.

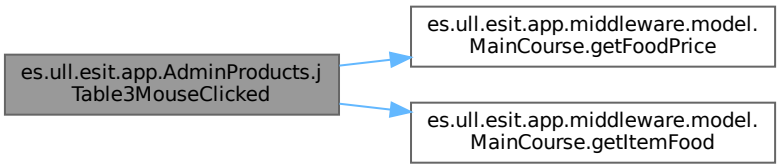
Definition at line 977 of file AdminProducts.java.

```
00977                                                                    {//
      GEN-FIRST:event_jTable3MouseClicked
00978     int row = jTable3.getSelectedRow();
00979     if (row == -1)
00980       return;
00981
00982     String idStr = jTable3.getValueAt(row, 0).toString();
00983     selectedMainCourseID = Long.valueOf(idStr);
00984     LOGGER.info("Selected Main Course ID: {}", selectedMainCourseID);
00985
```

```
00986     new Thread(() -> {
00987       try {
00988         MainCourse mainCourse = productService.getMainCourseById(selectedMainCourseID);
00989         SwingUtilities.invokeLater(() -> {
00990           itemname2.setText(mainCourse.getItemFood());
00991           itemprice2.setText(String.valueOf(mainCourse.getFoodPrice()));
00992         });
00993       } catch (Exception ex) {
00994         SwingUtilities.invokeLater(
00995             () -> JOptionPane.showMessageDialog(null, "Error loading main course details: " +
      ex.getMessage()));
00996       }
00997     }).start();
00998   }// GEN-LAST:event_jTable3MouseClicked
```

References es.ull.esit.app.middleware.model.MainCourse.getFoodPrice(), es.ull.esit.app.middleware.model.MainCourse.getItemFood
LOGGER, and productService.

Here is the call graph for this function:



9.3.3.13 main()

```
void es.ull.esit.app.AdminProducts.main (
             String[] args)  [inline], [static]
```

Standalone entry point for testing the AdminProducts window.

```
Sets the Nimbus look and feel if available and shows an instance
of AdminProducts. In the normal application flow, this window is
opened from AdminLogin after authentication.
```

Parameters

args	[String[]] Command line arguments (not used).

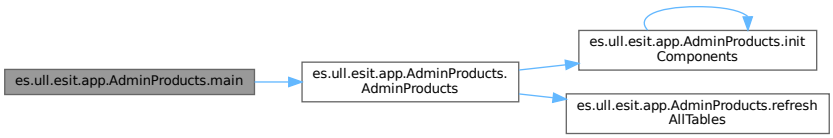
Definition at line 1096 of file AdminProducts.java.

```
01096                                       {
01097     try {
01098       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
01099         if ("Nimbus".equals(info.getName())) {
01100           javax.swing.UIManager.setLookAndFeel(info.getClassName());
01101           break;
01102         }
01103       }
01104     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
      javax.swing.UnsupportedLookAndFeelException ex) {
01105
      java.util.logging.Logger.getLogger(AdminProducts.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
```

```
01106     }
01107     // </editor-fold>
01108
01109     /* Create and display the form */
01110     java.awt.EventQueue.invokeLater(() -> new AdminProducts().setVisible(true));
01111   }
```

References AdminProducts().

Here is the call graph for this function:



9.3.3.14 refreshAllTables()

```
void es.ull.esit.app.AdminProducts.refreshAllTables ()  [inline], [private]
```

Reloads all product tables.

```
   Helper method that triggers asynchronous loading of drinks,
   appetizers and main courses from the server.
```

Definition at line 135 of file AdminProducts.java.

```
00135                                 {
00136     loadDrinks();
00137     loadAppetizer();
00138     loadmainCourse();
00139   }
```

Referenced by AdminProducts().

Here is the caller graph for this function:

9.3.4 Member Data Documentation

9.3.4.1 FIRST_COLUMN_HEADER

```
final String es.ull.esit.app.AdminProducts.FIRST_COLUMN_HEADER = "ID"  [static], [private]
```

Column header for the ID column in product tables.

Definition at line 41 of file AdminProducts.java.

9.3.4.2 LOGGER

```
final Logger es.ull.esit.app.AdminProducts.LOGGER = LoggerFactory.getLogger(AdminProducts.↩
class)  [static], [private]
```

Logger for logging events and errors.

Definition at line 33 of file AdminProducts.java.

Referenced by jTable1MouseClicked(), jTable2MouseClicked(), and jTable3MouseClicked().

9.3.4.3 PRIMARY_FONT

```
final String es.ull.esit.app.AdminProducts.PRIMARY_FONT = "Yu Gothic UI"  [static], [private]
```

Primary font used throughout the GUI.

Definition at line 36 of file AdminProducts.java.

Referenced by initComponents().

9.3.4.4 productService

```
final transient ProductService es.ull.esit.app.AdminProducts.productService  [private]
```

Service used to call the REST API and apply product-related logic.

Definition at line 30 of file AdminProducts.java.

Referenced by jButton1ActionPerformed(), jButton2ActionPerformed(), jButton4ActionPerformed(), jButton5ActionPerformed(), jButton6ActionPerformed(), jButton7ActionPerformed(), jTable1MouseClicked(), jTable2MouseClicked(), and jTable3MouseClicked().

9.3.4.5 SECOND_COLUMN_HEADER

```
final String es.ull.esit.app.AdminProducts.SECOND_COLUMN_HEADER = "Item Name"  [static], [private]
```

Column header for the name column in product tables.

Definition at line 43 of file AdminProducts.java.

Referenced by initComponents().

9.3.4.6 SECONDARY_FONT

```
final String es.ull.esit.app.AdminProducts.SECONDARY_FONT = "Thonburi"  [static], [private]
```

Secondary font used for buttons and smaller text.

Definition at line 38 of file AdminProducts.java.

9.3.4.7 THIRD_COLUMN_HEADER

```
final String es.ull.esit.app.AdminProducts.THIRD_COLUMN_HEADER = "Item Price"  [static], [private]
```

Column header for the price column in product tables.

Definition at line 45 of file AdminProducts.java.

Referenced by initComponents().

9.3.4.8 UPDATE_STRING

```
final String es.ull.esit.app.AdminProducts.UPDATE_STRING = "Update"  [static], [private]
```

String constant for the "Update" button label.

Definition at line 48 of file AdminProducts.java.

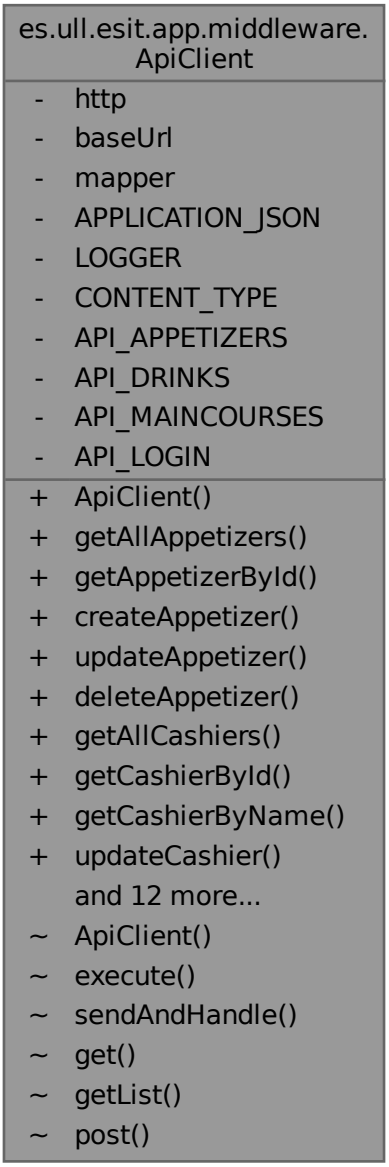
The documentation for this class was generated from the following file:

- AdminProducts.java

9.4 es.ull.esit.app.middleware.ApiClient Class Reference

API client for interacting with the backend REST API.

Collaboration diagram for es.ull.esit.app.middleware.ApiClient:



Classes

- interface IOCallable

  Functional interface for lambdas that may throw InterruptedException or IOException.
- interface ResponseHandler

  Functional interface for processing HTTP responses and possibly throwing IOException from JSON parsing.

Public Member Functions

- ApiClient (String baseUrl)

Constructs the ApiClient with the given base URL.

- List< Appetizer > getAllAppetizers ()

  GET all appetizers.
- Appetizer getAppetizerById (Long id)

  GET appetizer by ID.
- Appetizer createAppetizer (Appetizer appetizer)

  POST create new appetizer.
- Appetizer updateAppetizer (Long id, Appetizer appetizer)

  PUT update appetizer by ID.
- void deleteAppetizer (Long id)

  DELETE appetizer by ID.
- List< Cashier > getAllCashiers ()

  GET all cashiers.
- Cashier getCashierById (Long id)

  GET cashier by ID.
- Cashier getCashierByName (String name)

  GET cashier by name.
- Cashier updateCashier (Long id, Cashier cashier)

  PUT update cashier by ID.
- List< Drink > getAllDrinks ()

  GET all drinks.
- Drink getDrinkById (Long id)

  GET drink by ID.
- Drink createDrink (Drink drink)

  POST create new drink.
- Drink updateDrink (Long id, Drink drink)

  PUT update drink by ID.
- void deleteDrink (Long id)

  DELETE drink by ID.
- List< MainCourse > getAllMainCourses ()

  GET all maincourses.
- MainCourse getMainCourseById (Long id)

  GET maincourse by ID.
- MainCourse createMainCourse (MainCourse mainCourse)

  POST create new maincourse.
- MainCourse updateMainCourse (Long id, MainCourse mainCourse)

  PUT update maincourse by ID.
- void deleteMainCourse (Long id)

  DELETE maincourse by ID.
- void login (String ignored)

  Legacy login method kept for compatibility.
- User login (String username, String password)

  Authenticates a user against the backend.

Private Attributes

- final HttpClient http

  Low-level HTTP client to send requests.
- final String baseUrl

  Base URL of the REST API.
- final ObjectMapper mapper

  JSON object mapper for serialization/deserialization: converts between JSON and Java objects.

Static Private Attributes

- static final String APPLICATION_JSON = "application/json"

  Constant for JSON content type.
- static final Logger LOGGER = LoggerFactory.getLogger(ApiClient.class)

  Logger for logging events and errors.
- static final String CONTENT_TYPE = "Content-Type"

  HTTP header for content type.
- static final String API_APPETIZERS = "/api/appetizers/"

  API endpoint for appetizers.
- static final String API_DRINKS = "/api/drinks/"

  API endpoint for drinks.
- static final String API_MAINCOURSES = "/api/maincourses/"

  API endpoint for main courses.
- static final String API_LOGIN = "/api/login"

9.4.1 Detailed Description

API client for interacting with the backend REST API.

```
Wraps HttpClient and provides methods to:
- perform CRUD operations on Appetizers, Cashiers, Drinks, MainCourses.
- send login requests to the backend.
```

Definition at line 31 of file ApiClient.java.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 ApiClient()

```
es.ull.esit.app.middleware.ApiClient.ApiClient (
            String baseUrl)  [inline]
```

Constructs the ApiClient with the given base URL.

```
If the base URL ends with "/", the slash is removed to avoid double
slashes.
```

Parameters

baseUrl	[String] Base URL of the REST API, such as "http://localhost:8080".

Definition at line 74 of file ApiClient.java.

```
00074                                   {
00075     this.baseUrl = baseUrl.endsWith("/") ? baseUrl.substring(0, baseUrl.length() - 1) : baseUrl;
00076     this.http = HttpClient.newBuilder()
00077         .connectTimeout(Duration.ofSeconds(5))
00078         .build();
00079     this.mapper = new ObjectMapper();
00080   }
```

References baseUrl.

9.4.3 Member Function Documentation

9.4.3.1 createAppetizer()

```
Appetizer es.ull.esit.app.middleware.ApiClient.createAppetizer (
            Appetizer appetizer)  [inline]
```

POST create new appetizer.

Creates a new appetizer in the backend.

Parameters

appetizer	[Appetizer] Appetizer object to create.

Returns

[Appetizer] Created Appetizer object returned from the backend.

Definition at line 274 of file ApiClient.java.

```
00274                                                    {
00275     return post("/api/appetizers", appetizer, Appetizer.class);
00276   }
```

9.4.3.2 createDrink()

```
Drink es.ull.esit.app.middleware.ApiClient.createDrink (
            Drink drink)  [inline]
```

POST create new drink.

```
   Creates a new drink in the backend.
```

Parameters

drink	[Drink] Drink object to create.

Returns

[Drink] Created Drink object returned from the backend.

Definition at line 416 of file ApiClient.java.

```
00416                                          {
00417     return post("/api/drinks", drink, Drink.class);
00418   }
```

9.4.3.3 createMainCourse()


```
MainCourse es.ull.esit.app.middleware.ApiClient.createMainCourse (
            MainCourse mainCourse)  [inline]
```

POST create new maincourse.

Parameters

mainCourse	[MainCourse] MainCourse object to create.

Returns

[MainCourse] Created MainCourse object returned from the backend.

Definition at line 498 of file ApiClient.java.

```
00498                                                                    {
00499     return post("/api/maincourses", mainCourse, MainCourse.class);
00500   }
```

9.4.3.4 deleteAppetizer()

```
void es.ull.esit.app.middleware.ApiClient.deleteAppetizer (
             Long id)  [inline]
```

DELETE appetizer by ID.

Deletes an existing appetizer in the backend.

Parameters

id	[Long] ID of the appetizer to delete.

Definition at line 308 of file ApiClient.java.

```
00308                                       {
00309     String path = API_APPETIZERS + id;
00310     execute("deleteAppetizer id=" + id, () -> sendAndHandle("DELETE " + path,
00311         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).DELETE().build(), (status, body) -> {
00312           if (status < 200 || status >= 300) {
00313             throw new ApiClientException("DELETE " + path + " failed with HTTP " + status + " body: "
      + body);
00314           }
00315           return null;
00316         }));
00317   }
```

References API_APPETIZERS, and baseUrl.

9.4.3.5 deleteDrink()

```
void es.ull.esit.app.middleware.ApiClient.deleteDrink (
             Long id)  [inline]
```

DELETE drink by ID.

Deletes an existing drink in the backend.

Parameters

id	[Long] ID of the drink to delete.

Definition at line 452 of file ApiClient.java.

```
00452                                    {
00453     String path = API_DRINKS + id;
00454     execute("deleteDrink id=" + id, () -> sendAndHandle("DELETE " + path,
00455         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).DELETE().build(), (status, body) -> {
00456           if (status < 200 || status >= 300) {
00457             throw new ApiClientException("DELETE " + path + " failed with HTTP " + status + " body: "
      + body);
00458           }
00459           return null;
00460         }));
00461   }
```

References API_DRINKS, and baseUrl.

9.4.3.6 deleteMainCourse()

```
void es.ull.esit.app.middleware.ApiClient.deleteMainCourse (
             Long id)  [inline]
```

DELETE maincourse by ID.

```
   Deletes an existing maincourse in the backend by its ID.
```

Parameters

id	[Long] ID of the maincourse to delete.

Definition at line 534 of file ApiClient.java.

```
00534                                        {
00535     String path = API_MAINCOURSES + id;
00536     execute("deleteMainCourse id=" + id, () -> sendAndHandle("DELETE " + path,
00537         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).DELETE().build(), (status, body) -> {
00538           if (status < 200 || status >= 300) {
00539             throw new ApiClientException("DELETE " + path + " failed with HTTP " + status + " body: "
      + body);
00540           }
00541           return null;
00542         }));
00543   }
```

References API_MAINCOURSES, and baseUrl.

9.4.3.7 getAllAppetizers()

```
List< Appetizer > es.ull.esit.app.middleware.ApiClient.getAllAppetizers ()  [inline]
```

GET all appetizers.

```
   Retrieves a list of all appetizers from the backend.
```

Returns

[List<Appetizer>] List of all appetizers.

Definition at line 251 of file ApiClient.java.

```
00251                                        {
00252     return getList("/api/appetizers", new TypeReference<List<Appetizer>>() {});
00253   }
```

9.4.3.8 getAllCashiers()

```
List< Cashier > es.ull.esit.app.middleware.ApiClient.getAllCashiers ()  [inline]
```

GET all cashiers.

Retrieves a list of all cashiers from the backend.

Returns

[List<Cashier>] List of all cashiers.

Definition at line 330 of file ApiClient.java.

```
00330                                    {
00331     return getList("/api/cashiers", new TypeReference<List<Cashier»() {});
00332   }
```

9.4.3.9 getAllDrinks()

```
List< Drink > es.ull.esit.app.middleware.ApiClient.getAllDrinks ()  [inline]
```

GET all drinks.

```
Retrieves a list of all drinks from the backend.
```

Returns

[List<Drink>] List of all drinks.

Definition at line 392 of file ApiClient.java.

```
00392                                  {
00393     return getList("/api/drinks", new TypeReference<List<Drink»() {});
00394   }
```

9.4.3.10 getAllMainCourses()

```
List< MainCourse > es.ull.esit.app.middleware.ApiClient.getAllMainCourses ()  [inline]
```

GET all maincourses.

```
Retrieves a list of all maincourses from the backend.
```

Returns

[List<MainCourse>] List of all maincourses.

Definition at line 474 of file ApiClient.java.

```
00474                                        {
00475     return getList("/api/maincourses", new TypeReference<List<MainCourse»() {});
00476   }
```


9.4.3.11 getAppetizerById()

Parameters

id	[Long] ID of the appetizer.

Returns

[Appetizer] Appetizer object.

Definition at line 263 of file ApiClient.java.

```
00263                                           {
00264     return get(API_APPETIZERS + id, Appetizer.class);
00265   }
```

References API_APPETIZERS.

9.4.3.12 getCashierById()

```
Cashier es.ull.esit.app.middleware.ApiClient.getCashierById (
            Long id)  [inline]
```

GET cashier by ID.

Retrieves a cashier by its ID from the backend.

Parameters

id	[Long] ID of the cashier.

Returns

[Cashier] Cashier object.

Definition at line 341 of file ApiClient.java.

```
00341                                         {
00342     return get("/api/cashiers/" + id, Cashier.class);
00343   }
```

9.4.3.13 getCashierByName()

```
Cashier es.ull.esit.app.middleware.ApiClient.getCashierByName (
            String name)  [inline]
```

GET cashier by name.

Retrieves a cashier by its username from the backend.

Parameters

name	[String] username of the cashier.

Returns

[Cashier] Cashier object.

Definition at line 352 of file ApiClient.java.

```
00352                                                {
00353     return get("/api/cashiers/name/" + name, Cashier.class);
00354   }
```

9.4.3.14 getDrinkById()

```
Drink es.ull.esit.app.middleware.ApiClient.getDrinkById (
            Long id)  [inline]
```

GET drink by ID.

Retrieves a drink by its ID from the backend.

Parameters

id	[Long] ID of the drink.

Returns

[Drink] Drink object.

Definition at line 404 of file ApiClient.java.

```
00404                                        {
00405     return get(API_DRINKS + id, Drink.class);
00406   }
```

References API_DRINKS.

9.4.3.15 getMainCourseById()

```
MainCourse es.ull.esit.app.middleware.ApiClient.getMainCourseById (
            Long id)  [inline]
```

GET maincourse by ID.

Retrieves a maincourse by its ID from the backend.

Parameters

id	[Long] ID of the maincourse.

Returns

[MainCourse] MainCourse object.

Definition at line 486 of file ApiClient.java.

```
00486                                                {
00487     return get(API_MAINCOURSES + id, MainCourse.class);
00488   }
```

References API_MAINCOURSES.

9.4.3.16 login() [1/2]

```
void es.ull.esit.app.middleware.ApiClient.login (
            String ignored)  [inline]
```

Legacy login method kept for compatibility.

```
If authentication is added in the future, it can be implemented here.
```

Parameters

ignored	[String] Ignored parameter.

Definition at line 556 of file ApiClient.java.

```
00556                                    {
00557     // No-op: kept for compatibility.
00558   }
```

9.4.3.17 login() [2/2]

```
User es.ull.esit.app.middleware.ApiClient.login (
            String username,
            String password)  [inline]
```

Authenticates a user against the backend.

```
Sends a POST request to "/api/login" with username and password.
If the response status is 200, it returns the User parsed from JSON.
For any other status code it throws ApiClientException.
```

Parameters

username	[String] Username entered by the user.
password	[String] Password entered by the user.

Returns

[User] Authenticated User object.

Definition at line 571 of file ApiClient.java.

```
00571                                                 {
00572     String path = API_LOGIN;
00573     return execute("login for user=" + username, () -> {
00574       String jsonBody = mapper.writeValueAsString(Map.of(
00575           "username", username,
00576           "password", password));
00577
00578       HttpRequest request = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00579           .header(CONTENT_TYPE,
      APPLICATION_JSON).POST(HttpRequest.BodyPublishers.ofString(jsonBody)).build();
00580
00581       return sendAndHandle("POST " + path, request, (status, body) -> {
00582         if (status == 200) {
00583           return mapper.readValue(body, User.class);
00584         } else {
00585           throw new ApiClientException("Login failed with status: " + status + " body: " + body);
00586         }
00587       });
00588     });
00589   }
```

References API_LOGIN, APPLICATION_JSON, baseUrl, CONTENT_TYPE, and mapper.

9.4.3.18 updateAppetizer()

```
Appetizer es.ull.esit.app.middleware.ApiClient.updateAppetizer (
            Long id,
            Appetizer appetizer)  [inline]
```

PUT update appetizer by ID.

Updates an existing appetizer in the backend.

Parameters

id	[Long] ID of the appetizer to update.
appetizer	[Appetizer] Appetizer object with updated data.

Returns

[Appetizer] Updated Appetizer object returned from the backend.

Definition at line 286 of file ApiClient.java.

```
00286                                                         {
00287     String path = API_APPETIZERS + id;
00288     return execute("updateAppetizer id=" + id, () -> {
00289       String json = mapper.writeValueAsString(appetizer);
00290       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00291           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
      APPLICATION_JSON).build();
00292
00293       return sendAndHandle("PUT " + path, req, (status, body) -> {
00294         if (status < 200 || status >= 300) {
00295           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
      body);
00296         }
00297         return mapper.readValue(body, Appetizer.class);
00298       });
00299     });
00300   }
```

References API_APPETIZERS, APPLICATION_JSON, baseUrl, CONTENT_TYPE, and mapper.

9.4.3.19 updateCashier()

```
Cashier es.ull.esit.app.middleware.ApiClient.updateCashier (
            Long id,
            Cashier cashier)  [inline]
```

PUT update cashier by ID.

Updates an existing cashier in the backend (name and/or salary).

Parameters

id	[Long] ID of the cashier to update.
cashier	[Cashier] Cashier object with updated data.

Returns

[Cashier] Updated Cashier object returned from the backend.

Definition at line 365 of file ApiClient.java.

```
00365                                                    {
00366     String path = "/api/cashiers/" + id;
00367     return execute("updateCashier id=" + id, () -> {
00368       String json = mapper.writeValueAsString(cashier);
00369       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00370           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
      APPLICATION_JSON).build();
00371
00372       return sendAndHandle("PUT " + path, req, (status, body) -> {
00373         if (status < 200 || status >= 300) {
00374           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
      body);
00375         }
00376         return mapper.readValue(body, Cashier.class);
00377       });
00378     });
00379   }
```

References APPLICATION_JSON, baseUrl, CONTENT_TYPE, and mapper.

9.4.3.20 updateDrink()

```
Drink es.ull.esit.app.middleware.ApiClient.updateDrink (
            Long id,
            Drink drink)  [inline]
```

PUT update drink by ID.

Updates an existing drink in the backend by its ID.

Parameters

id	[Long] ID of the drink to update.

drink	[Drink] Drink object with updated data.

Returns

[Drink] Updated Drink object returned from the backend.

Definition at line 429 of file ApiClient.java.

```
00429                                                {
00430     String path = API_DRINKS + id;
00431     return execute("updateDrink id=" + id, () -> {
00432       String json = mapper.writeValueAsString(drink);
00433       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00434           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
      APPLICATION_JSON).build();
00435
00436       return sendAndHandle("PUT " + path, req, (status, body) -> {
00437         if (status < 200 || status >= 300) {
00438           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
      body);
00439         }
00440         return mapper.readValue(body, Drink.class);
00441       });
00442     });
00443   }
```

References API_DRINKS, APPLICATION_JSON, baseUrl, CONTENT_TYPE, and mapper.

9.4.3.21 updateMainCourse()

```
MainCourse es.ull.esit.app.middleware.ApiClient.updateMainCourse (
            Long id,
            MainCourse mainCourse)  [inline]
```

PUT update maincourse by ID.

```
Updates an existing maincourse in the backend.
```

Parameters

id	[Long] ID of the maincourse to update.
mainCourse	[MainCourse] MainCourse object with updated data.

Returns

[MainCourse] Updated MainCourse object returned from the backend.

Definition at line 511 of file ApiClient.java.

```
00511                                                          {
00512     String path = API_MAINCOURSES + id;
00513     return execute("updateMainCourse id=" + id, () -> {
00514       String json = mapper.writeValueAsString(mainCourse);
00515       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00516           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
      APPLICATION_JSON).build();
00517
00518       return sendAndHandle("PUT " + path, req, (status, body) -> {
00519         if (status < 200 || status >= 300) {
00520           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
      body);
00521         }
00522         return mapper.readValue(body, MainCourse.class);
00523       });
00524     });
00525   }
```

References API_MAINCOURSES, APPLICATION_JSON, baseUrl, CONTENT_TYPE, and mapper.

9.4.4 Member Data Documentation

9.4.4.1 API_APPETIZERS

```
final String es.ull.esit.app.middleware.ApiClient.API_APPETIZERS = "/api/appetizers/"  [static], [private]
```

API endpoint for appetizers.

Definition at line 43 of file ApiClient.java.

Referenced by deleteAppetizer(), getAppetizerById(), and updateAppetizer().

9.4.4.2 API_DRINKS

```
final String es.ull.esit.app.middleware.ApiClient.API_DRINKS = "/api/drinks/"  [static], [private]
```

API endpoint for drinks.

Definition at line 46 of file ApiClient.java.

Referenced by deleteDrink(), getDrinkById(), and updateDrink().

9.4.4.3 API_LOGIN

```
final String es.ull.esit.app.middleware.ApiClient.API_LOGIN = "/api/login"  [static], [private]
```

Definition at line 51 of file ApiClient.java.

Referenced by login().

9.4.4.4 API_MAINCOURSES

```
final String es.ull.esit.app.middleware.ApiClient.API_MAINCOURSES = "/api/maincourses/"  [static], [private]
```

API endpoint for main courses.

Definition at line 49 of file ApiClient.java.

Referenced by deleteMainCourse(), getMainCourseById(), and updateMainCourse().

9.4.4.5 APPLICATION_JSON

```
final String es.ull.esit.app.middleware.ApiClient.APPLICATION_JSON = "application/json"  [static], [private]
```

Constant for JSON content type.

Definition at line 34 of file ApiClient.java.

Referenced by login(), updateAppetizer(), updateCashier(), updateDrink(), and updateMainCourse().

9.4.4.6 baseUrl

```
final String es.ull.esit.app.middleware.ApiClient.baseUrl  [private]
```

Base URL of the REST API.

Definition at line 57 of file ApiClient.java.

Referenced by ApiClient(), deleteAppetizer(), deleteDrink(), deleteMainCourse(), login(), updateAppetizer(), updateCashier(), updateDrink(), and updateMainCourse().

9.4.4.7 CONTENT_TYPE

```
final String es.ull.esit.app.middleware.ApiClient.CONTENT_TYPE = "Content-Type"  [static],
[private]
```

HTTP header for content type.

Definition at line 40 of file ApiClient.java.

Referenced by login(), updateAppetizer(), updateCashier(), updateDrink(), and updateMainCourse().

9.4.4.8 http

```
final HttpClient es.ull.esit.app.middleware.ApiClient.http  [private]
```

Low-level HTTP client to send requests.

Definition at line 54 of file ApiClient.java.

9.4.4.9 LOGGER

```
final Logger es.ull.esit.app.middleware.ApiClient.LOGGER = LoggerFactory.getLogger(ApiClient.↩
class)  [static], [private]
```

Logger for logging events and errors.

Definition at line 37 of file ApiClient.java.

9.4.4.10 mapper

```
final ObjectMapper es.ull.esit.app.middleware.ApiClient.mapper  [private]
```

JSON object mapper for serialization/deserialization: converts between JSON and Java objects.

Definition at line 63 of file ApiClient.java.

Referenced by login(), updateAppetizer(), updateCashier(), updateDrink(), and updateMainCourse().

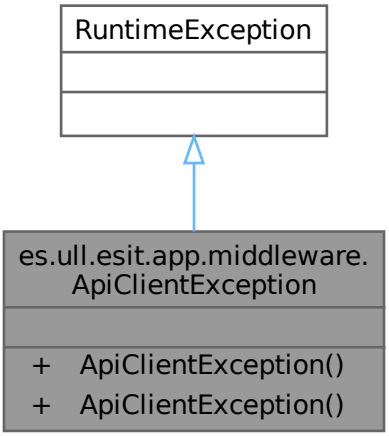
The documentation for this class was generated from the following file:

- ApiClient.java

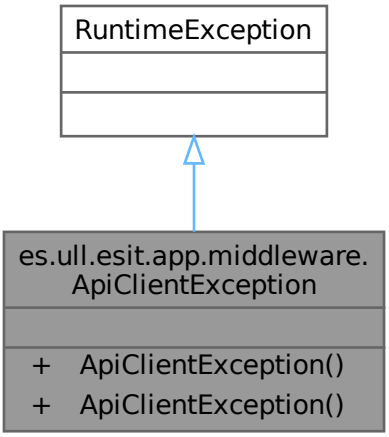
9.5 es.ull.esit.app.middleware.ApiClientException Class Reference

Custom exception class for API client errors.

Inheritance diagram for es.ull.esit.app.middleware.ApiClientException:



Collaboration diagram for es.ull.esit.app.middleware.ApiClientException:



Public Member Functions

- ApiClientException (String message)

  Constructs a new ApiClientException with the specified detail message.
- ApiClientException (String message, Throwable cause)

  Constructs a new ApiClientException with the specified detail message and cause.

9.5.1 Detailed Description

Custom exception class for API client errors.

Inherits from RuntimeException to represent exceptions that occur during API client operations.

Definition at line 9 of file ApiClientException.java.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 ApiClientException() [1/2]

```
es.ull.esit.app.middleware.ApiClientException.ApiClientException (
            String message)  [inline]
```

Constructs a new ApiClientException with the specified detail message.

Parameters

message	[String] Detail message for the exception.

Definition at line 16 of file ApiClientException.java.

```
00016                                           {
00017     super(message);
00018   }
```

9.5.2.2 ApiClientException() [2/2]

```
es.ull.esit.app.middleware.ApiClientException.ApiClientException (
            String message,
            Throwable cause)  [inline]
```

Constructs a new ApiClientException with the specified detail message and cause.

Parameters

message	[String] Detail message for the exception.
cause	[Throwable] The cause of the exception.

Definition at line 26 of file ApiClientException.java.

```
00026                                                       {
00027     super(message, cause);
00028   }
```

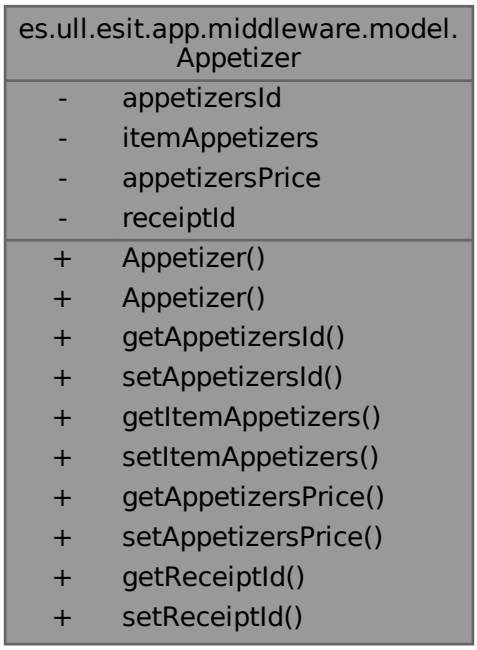
The documentation for this class was generated from the following file:

- ApiClientException.java

9.6 es.ull.esit.app.middleware.model.Appetizer Class Reference

Client-side model representing an appetizer received from the backend API.

Collaboration diagram for es.ull.esit.app.middleware.model.Appetizer:



Public Member Functions

- Appetizer ()
  Default constructor required for JSON deserialization.
- Appetizer (Long appetizersId, String itemAppetizers, Integer appetizersPrice, Long receiptId)
  Constructs an appetizer with all fields.
- Long getAppetizersId ()
  Gets the appetizer identifier.
- void setAppetizersId (Long appetizersId)
  Sets the appetizer identifier.
- String getItemAppetizers ()
  Gets the appetizer item name.
- void setItemAppetizers (String itemAppetizers)
  Sets the appetizer item name.
- Integer getAppetizersPrice ()
  Gets the appetizer price.
- void setAppetizersPrice (Integer appetizersPrice)
  Sets the appetizer price.
- Long getReceiptId ()
  Gets the identifier of the related receipt.
- void setReceiptId (Long receiptId)
  Sets the identifier of the related receipt.

Private Attributes

- Long appetizersId

  Unique identifier of the appetizer (JSON property "appetizersId").
- String itemAppetizers

  Name of the appetizer item (JSON property "itemAppetizers").
- Integer appetizersPrice

  Price of the appetizer (JSON property "appetizersPrice").
- Long receiptId

  Identifier of the receipt this appetizer belongs to (JSON property "receiptId").

9.6.1 Detailed Description

Client-side model representing an appetizer received from the backend API.

```
It is used by the Swing application to deserialize JSON sent by
the REST server for the "/api/appetizers" endpoint.
```

Definition at line 11 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 Appetizer() [1/2]

```
es.ull.esit.app.middleware.model.Appetizer.Appetizer ()  [inline]
```

Default constructor required for JSON deserialization.

Definition at line 35 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00035                     {
00036     }
```

References Appetizer().

Referenced by Appetizer().

Here is the call graph for this function:

Here is the caller graph for this function:

9.6.2.2 Appetizer() [2/2]

```
es.ull.esit.app.middleware.model.Appetizer.Appetizer (
            Long appetizersId,
            String itemAppetizers,
            Integer appetizersPrice,
            Long receiptId)  [inline]
```

Constructs an appetizer with all fields.

Parameters

appetizersId	[Long] Unique identifier of the appetizer.
itemAppetizers	[String] Name of the appetizer item.
appetizersPrice	[Integer] Price of the appetizer.
receiptId	[Long] Identifier of the related receipt (optional).

Definition at line 46 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00046
    {
00047     this.appetizersId = appetizersId;
00048     this.itemAppetizers = itemAppetizers;
00049     this.appetizersPrice = appetizersPrice;
00050     this.receiptId = receiptId;
00051   }
```

References appetizersId, appetizersPrice, itemAppetizers, and receiptId.

9.6.3 Member Function Documentation

9.6.3.1 getAppetizersId()

```
Long es.ull.esit.app.middleware.model.Appetizer.getAppetizersId ()  [inline]
```

Gets the appetizer identifier.

Returns

[Long] Unique identifier of the appetizer.

Definition at line 58 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00058                                {
00059     return appetizersId;
00060   }
```

References appetizersId.

9.6.3.2 getAppetizersPrice()

```
Integer es.ull.esit.app.middleware.model.Appetizer.getAppetizersPrice ()  [inline]
```

Gets the appetizer price.

Returns

[Integer] Price of the appetizer.

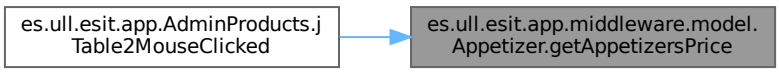
Definition at line 94 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00094                                    {
00095     return appetizersPrice;
00096   }
```

References appetizersPrice.

Referenced by es.ull.esit.app.AdminProducts.jTable2MouseClicked().

Here is the caller graph for this function:



9.6.3.3 getItemAppetizers()

```
String es.ull.esit.app.middleware.model.Appetizer.getItemAppetizers ()  [inline]
```

Gets the appetizer item name.

Returns

[String] Name of the appetizer item.

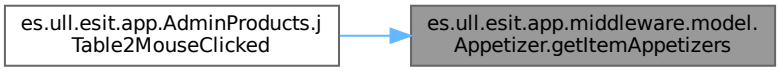
Definition at line 76 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00076                                    {
00077     return itemAppetizers;
00078   }
```

References itemAppetizers.

Referenced by es.ull.esit.app.AdminProducts.jTable2MouseClicked().

Here is the caller graph for this function:



9.6.3.4 getReceiptId()

```
Long es.ull.esit.app.middleware.model.Appetizer.getReceiptId ()  [inline]
```

Gets the identifier of the related receipt.

Returns

[Long] Identifier of the receipt or null if not set.

Definition at line 112 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00112                                {
00113     return receiptId;
00114   }
```

References receiptId.

9.6.3.5 setAppetizersId()

```
void es.ull.esit.app.middleware.model.Appetizer.setAppetizersId (
            Long appetizersId)  [inline]
```

Sets the appetizer identifier.

Parameters

appetizersId	[Long] Unique identifier of the appetizer.

Definition at line 67 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00067                                                {
00068     this.appetizersId = appetizersId;
00069   }
```

References appetizersId.

9.6.3.6 setAppetizersPrice()

```
void es.ull.esit.app.middleware.model.Appetizer.setAppetizersPrice (
            Integer appetizersPrice)  [inline]
```

Sets the appetizer price.

Parameters

appetizersPrice	[Integer] Price of the appetizer.

Definition at line 103 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00103                                                    {
00104     this.appetizersPrice = appetizersPrice;
00105   }
```

References appetizersPrice.

9.6.3.7 setItemAppetizers()

```
void es.ull.esit.app.middleware.model.Appetizer.setItemAppetizers (
            String itemAppetizers)  [inline]
```

Sets the appetizer item name.

Parameters

itemAppetizers	[String] Name of the appetizer item.

Definition at line 85 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00085                                                {
00086     this.itemAppetizers = itemAppetizers;
00087   }
```

References itemAppetizers.

9.6.3.8 setReceiptId()

```
void es.ull.esit.app.middleware.model.Appetizer.setReceiptId (
            Long receiptId)  [inline]
```

Sets the identifier of the related receipt.

Parameters

receipt↩ Id	[Long] Identifier of the receipt.

Definition at line 121 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

```
00121                                          {
00122     this.receiptId = receiptId;
00123   }
```

References receiptId.

9.6.4 Member Data Documentation

9.6.4.1 appetizersId

```
Long es.ull.esit.app.middleware.model.Appetizer.appetizersId  [private]
```

Unique identifier of the appetizer (JSON property "appetizersId").

Definition at line 15 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

Referenced by Appetizer(), getAppetizersId(), and setAppetizersId().

9.6.4.2 appetizersPrice

```
Integer es.ull.esit.app.middleware.model.Appetizer.appetizersPrice  [private]
```

Price of the appetizer (JSON property "appetizersPrice").

Definition at line 23 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

Referenced by Appetizer(), getAppetizersPrice(), and setAppetizersPrice().

9.6.4.3 itemAppetizers

```
String es.ull.esit.app.middleware.model.Appetizer.itemAppetizers  [private]
```

Name of the appetizer item (JSON property "itemAppetizers").

Definition at line 19 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

Referenced by Appetizer(), getItemAppetizers(), and setItemAppetizers().

9.6.4.4 receiptId

```
Long es.ull.esit.app.middleware.model.Appetizer.receiptId  [private]
```

Identifier of the receipt this appetizer belongs to (JSON property "receiptId").

Currently not populated by the backend.

Definition at line 30 of file src/main/java/es/ull/esit/app/middleware/model/Appetizer.java.

Referenced by Appetizer(), getReceiptId(), and setReceiptId().

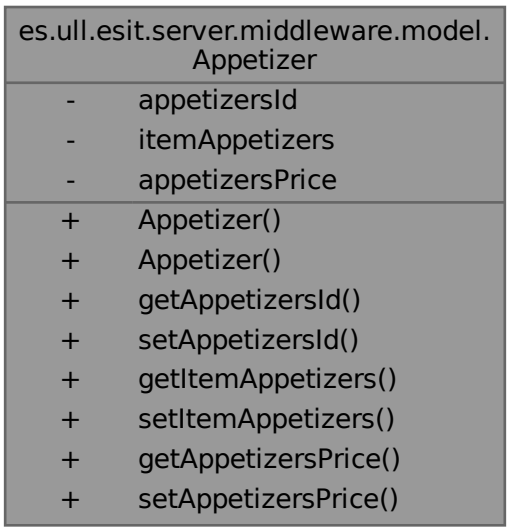
The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/app/middleware/model/Appetizer.java

9.7 es.ull.esit.server.middleware.model.Appetizer Class Reference

JPA entity that represents an appetizer in the menu.

Collaboration diagram for es.ull.esit.server.middleware.model.Appetizer:



Public Member Functions

- Appetizer ()

  Default constructor required by JPA.
- Appetizer (Long appetizersId, String itemAppetizers, Integer appetizersPrice)

  Full constructor.
- Long getAppetizersId ()

  Gets the appetizer ID.
- void setAppetizersId (Long appetizersId)

  Sets the appetizer ID.
- String getItemAppetizers ()

  Gets the appetizer name.
- void setItemAppetizers (String itemAppetizers)

  Sets the appetizer name.
- Integer getAppetizersPrice ()

  Gets the appetizer price.
- void setAppetizersPrice (Integer appetizersPrice)

  Sets the appetizer price.

Private Attributes

- Long appetizersId

  Primary key of the appetizer (column "id").
- String itemAppetizers

  Name of the appetizer (column "name").
- Integer appetizersPrice

  Price of the appetizer (column "price").

9.7.1 Detailed Description

JPA entity that represents an appetizer in the menu.

```
It is mapped to the "appetizers" table used by the REST API.
Each record has an auto-increment ID, a name and a price.
```

Definition at line 15 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 Appetizer() [1/2]

```
es.ull.esit.server.middleware.model.Appetizer.Appetizer ()  [inline]
```

Default constructor required by JPA.

Definition at line 34 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00034                     {
00035     }
```

References Appetizer().

Referenced by Appetizer().

Here is the call graph for this function:

Here is the caller graph for this function:

9.7.2.2 Appetizer() [2/2]

```
es.ull.esit.server.middleware.model.Appetizer.Appetizer (
            Long appetizersId,
            String itemAppetizers,
            Integer appetizersPrice)  [inline]
```

Full constructor.

Parameters

appetizersId	[Long] Identifier of the appetizer.
itemAppetizers	[String] Name of the appetizer.
appetizersPrice	[Integer] Price of the appetizer.

Definition at line 44 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00044                                                                                {
00045     this.appetizersId = appetizersId;
00046     this.itemAppetizers = itemAppetizers;
00047     this.appetizersPrice = appetizersPrice;
00048   }
```

References appetizersId, appetizersPrice, and itemAppetizers.

9.7.3 Member Function Documentation

9.7.3.1 getAppetizersId()

```
Long es.ull.esit.server.middleware.model.Appetizer.getAppetizersId ()  [inline]
```

Gets the appetizer ID.

Returns

[Long] Appetizer id.

Definition at line 55 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00055                                {
00056     return appetizersId;
00057   }
```

References appetizersId.

9.7.3.2 getAppetizersPrice()

```
Integer es.ull.esit.server.middleware.model.Appetizer.getAppetizersPrice ()  [inline]
```

Gets the appetizer price.

Returns

[Integer] Appetizer price.

Definition at line 92 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00092                                    {
00093     return appetizersPrice;
00094   }
```

References appetizersPrice.

9.7.3.3 getItemAppetizers()

```
String es.ull.esit.server.middleware.model.Appetizer.getItemAppetizers ()  [inline]
```

Gets the appetizer name.

Returns

[String] Appetizer name.

Definition at line 74 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00074                                   {
00075     return itemAppetizers;
00076   }
```

References itemAppetizers.

9.7.3.4 setAppetizersId()

```
void es.ull.esit.server.middleware.model.Appetizer.setAppetizersId (
            Long appetizersId)  [inline]
```

Sets the appetizer ID.

Generated by the database.

Parameters

appetizersId	[Long] Appetizer id.

Definition at line 65 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00065                                           {
00066     this.appetizersId = appetizersId;
00067   }
```


References appetizersId.

Parameters

appetizersPrice	[Integer] Appetizer price.

Definition at line 101 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00101                                                    {
00102     this.appetizersPrice = appetizersPrice;
00103   }
```

References appetizersPrice.

9.7.3.6 setItemAppetizers()

```
void es.ull.esit.server.middleware.model.Appetizer.setItemAppetizers (
             String itemAppetizers)  [inline]
```

Sets the appetizer name.

Parameters

itemAppetizers	[String] Appetizer name.

Definition at line 83 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

```
00083                                                {
00084     this.itemAppetizers = itemAppetizers;
00085   }
```

References itemAppetizers.

9.7.4 Member Data Documentation

9.7.4.1 appetizersId

```
Long es.ull.esit.server.middleware.model.Appetizer.appetizersId  [private]
```

Primary key of the appetizer (column "id").

Definition at line 21 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

Referenced by Appetizer(), getAppetizersId(), and setAppetizersId().

9.7.4.2 appetizersPrice

```
Integer es.ull.esit.server.middleware.model.Appetizer.appetizersPrice  [private]
```

Price of the appetizer (column "price").

Definition at line 29 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

Referenced by Appetizer(), getAppetizersPrice(), and setAppetizersPrice().

9.7.4.3 itemAppetizers

```
String es.ull.esit.server.middleware.model.Appetizer.itemAppetizers  [private]
```

Name of the appetizer (column "name").

Definition at line 25 of file server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java.

Referenced by Appetizer(), getItemAppetizers(), and setItemAppetizers().

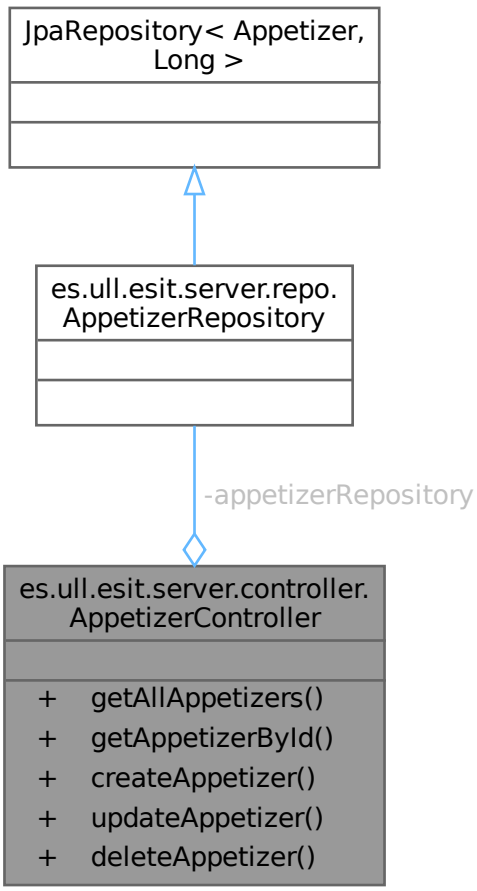
The documentation for this class was generated from the following file:

- server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java

9.8 es.ull.esit.server.controller.AppetizerController Class Reference

REST controller for managing appetizers.

Collaboration diagram for es.ull.esit.server.controller.AppetizerController:



Public Member Functions

- ResponseEntity< List< Appetizer > > getAllAppetizers ()

  Returns the complete list of appetizers.
- ResponseEntity< Appetizer > getAppetizerById (@PathVariable Long id)

  Returns a single appetizer by its id.
- ResponseEntity< Appetizer > createAppetizer (@RequestBody Appetizer appetizer)

  Creates a new appetizer.
- ResponseEntity< Appetizer > updateAppetizer (@PathVariable Long id, @RequestBody Appetizer appetizer)

  Updates an existing appetizer.
- ResponseEntity< Void > deleteAppetizer (@PathVariable Long id)

  Deletes an appetizer by its id.

Private Attributes

- AppetizerRepository appetizerRepository

  Repository used to perform database operations on appetizers.

9.8.1 Detailed Description

REST controller for managing appetizers.

Rovides CRUD operations for Appetizer entities using the path "/api/appetizers". These endpoints are used by the Swing client to load and modify appetizer information.

Definition at line 21 of file AppetizerController.java.

9.8.2 Member Function Documentation

9.8.2.1 createAppetizer()

```
ResponseEntity< Appetizer > es.ull.esit.server.controller.AppetizerController.createAppetizer
(
            @RequestBody Appetizer appetizer)  [inline]
```

Creates a new appetizer.

```
HTTP POST /api/appetizers
```

Parameters

appetizer	[Appetizer] Appetizer to create.

Returns

[ResponseEntity<Appetizer>] The created appetizer with status 201.

Definition at line 65 of file AppetizerController.java.

```
00065                                                                          {
00066     Appetizer saved = appetizerRepository.save(appetizer);
00067     return ResponseEntity.status(HttpStatus.CREATED).body(saved);
00068   }
```

References appetizerRepository.

9.8.2.2 deleteAppetizer()

```
ResponseEntity< Void > es.ull.esit.server.controller.AppetizerController.deleteAppetizer (
            @PathVariable Long id)  [inline]
```

Deletes an appetizer by its id.

```
HTTP DELETE /api/appetizers/{id}
```

Parameters

id	[Long] Identifier of the appetizer to delete.

Returns

[ResponseEntity<Void>] 204 if deleted or 404 if not found.

Definition at line 99 of file AppetizerController.java.

```
00099                                                                    {
00100     if (!appetizerRepository.existsById(id)) {
00101       return ResponseEntity.notFound().build();
00102     }
00103     appetizerRepository.deleteById(id);
00104     return ResponseEntity.noContent().build();
00105   }
```

References appetizerRepository, and deleteAppetizer().

Referenced by deleteAppetizer().

Here is the call graph for this function:

Here is the caller graph for this function:

9.8.2.3 getAllAppetizers()

```
ResponseEntity< List< Appetizer > > es.ull.esit.server.controller.AppetizerController.get↩
AllAppetizers ()  [inline]
```

Returns the complete list of appetizers.

```
HTTP GET /api/appetizers
```

Returns

[ResponseEntity<List<Appetizer>>] List of appetizers with status 200.

Definition at line 35 of file AppetizerController.java.

```
00035                                                          {
00036     List<Appetizer> appetizers = appetizerRepository.findAll();
00037     return ResponseEntity.ok(appetizers);
00038   }
```

References appetizerRepository.

9.8.2.4 getAppetizerById()

```
ResponseEntity< Appetizer > es.ull.esit.server.controller.AppetizerController.getAppetizerById
(
             @PathVariable Long id)  [inline]
```

Returns a single appetizer by its id.

```
HTTP GET /api/appetizers/{id}
```

Parameters

id	[Long] Identifier of the appetizer.

Returns

[ResponseEntity<Appetizer>] The appetizer if found or 404 otherwise.

Definition at line 49 of file AppetizerController.java.

```
00049                                                                  {
00050     Optional<Appetizer> appetizer = appetizerRepository.findById(id);
00051     return appetizer
00052         .map(ResponseEntity::ok)
00053         .orElseGet(() -> ResponseEntity.notFound().build());
00054   }
```

References appetizerRepository, and getAppetizerById().

Referenced by getAppetizerById().

Here is the call graph for this function:

Here is the caller graph for this function:

9.8.2.5 updateAppetizer()

```
ResponseEntity< Appetizer > es.ull.esit.server.controller.AppetizerController.updateAppetizer
(
            @PathVariable Long id,
            @RequestBody Appetizer appetizer)  [inline]
```

Updates an existing appetizer.

```
HTTP PUT /api/appetizers/{id}
```

Parameters

id	[Long] Identifier of the appetizer to update.
appetizer	[Appetizer] Updated appetizer data.

Returns

[ResponseEntity<Appetizer>] The updated appetizer or 404 if not found.

Definition at line 80 of file AppetizerController.java.

```
00081                                                                                {
00082     if (!appetizerRepository.existsById(id)) {
00083       return ResponseEntity.notFound().build();
00084     }
00085     appetizer.setAppetizersId(id);
00086     Appetizer updated = appetizerRepository.save(appetizer);
00087     return ResponseEntity.ok(updated);
00088   }
```

References appetizerRepository, and updateAppetizer().

Referenced by updateAppetizer().

Here is the call graph for this function:

Here is the caller graph for this function:

9.8.3 Member Data Documentation

9.8.3.1 appetizerRepository

```
AppetizerRepository es.ull.esit.server.controller.AppetizerController.appetizerRepository
[private]
```

Repository used to perform database operations on appetizers.

Definition at line 25 of file AppetizerController.java.

Referenced by createAppetizer(), deleteAppetizer(), getAllAppetizers(), getAppetizerById(), and updateAppetizer().

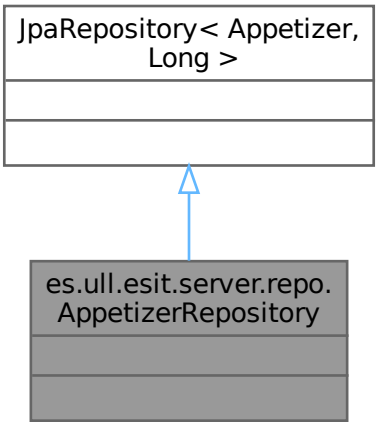
The documentation for this class was generated from the following file:

- AppetizerController.java

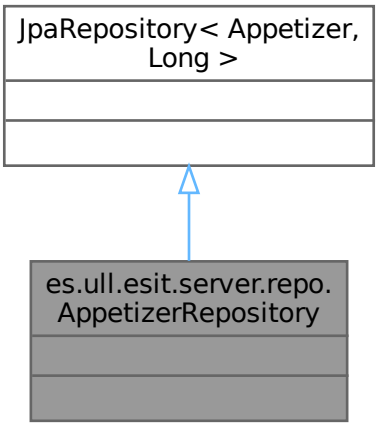
9.9 es.ull.esit.server.repo.AppetizerRepository Interface Reference

Repository interface for managing appetizers in the database.

Inheritance diagram for es.ull.esit.server.repo.AppetizerRepository:



Collaboration diagram for es.ull.esit.server.repo.AppetizerRepository:



9.9.1 Detailed Description

Repository interface for managing appetizers in the database.

```
Extends JpaRepository to provide basic CRUD operations on the "appetizers"
table, such as findAll, findById, save and delete.
```

Definition at line 12 of file AppetizerRepository.java.

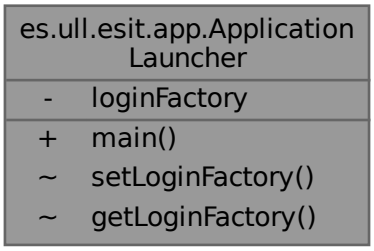
The documentation for this interface was generated from the following file:

- AppetizerRepository.java

9.10 es.ull.esit.app.ApplicationLauncher Class Reference

Auxiliary entry point used to start the application's UI.

Collaboration diagram for es.ull.esit.app.ApplicationLauncher:



Static Public Member Functions

- static void main (String[ ] args)

  Alternate entry point used to launch the Login window.

Static Private Attributes

- static java.util.function.Supplier< Login > loginFactory = Login::new

9.10.1 Detailed Description

Auxiliary entry point used to start the application's UI.

```
Serves as a simple launcher responsible for opening
the Login window when the program is executed directly.
```

Definition at line 10 of file ApplicationLauncher.java.

9.10.2 Member Function Documentation

Parameters

args	[String[]] Command-line arguments (unused).

Definition at line 19 of file ApplicationLauncher.java.

```
00019                                       {
00020     // Use a factory so tests can inject a mock Login instance and avoid
00021     // creating real GUI components (which can fail in headless CI).
00022     java.awt.EventQueue.invokeLater(() -> getLoginFactory().get().setVisible(true));
00023   }
```

9.10.3 Member Data Documentation

9.10.3.1 loginFactory

```
java.util.function.Supplier<Login> es.ull.esit.app.ApplicationLauncher.loginFactory = Login↩
::new  [static], [private]
```

Definition at line 30 of file ApplicationLauncher.java.

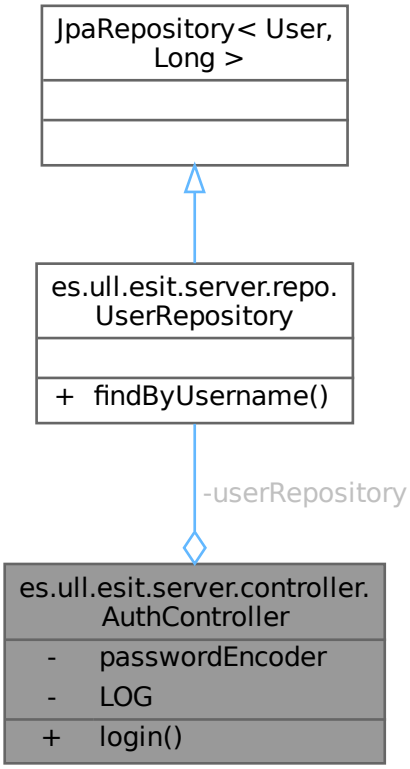
The documentation for this class was generated from the following file:

- ApplicationLauncher.java

9.11 es.ull.esit.server.controller.AuthController Class Reference

Rest controller for handling authentication-related requests.

Collaboration diagram for es.ull.esit.server.controller.AuthController:



Classes

- class LoginRequest

  Simple DTO (Data Transfer Object) for login requests payload.

Public Member Functions

- ResponseEntity<?> login (@RequestBody LoginRequest request)

  Endpoint for handling user login requests:

Private Attributes

- UserRepository userRepository

  Repository for accessing user data.
- PasswordEncoder passwordEncoder

  Component used to verify raw passwords against stored hashes.

Static Private Attributes

- static final Logger LOG = LoggerFactory.getLogger(AuthController.class)

  Logger for logging authentication events.

9.11.1 Detailed Description

Rest controller for handling authentication-related requests.

```
Exposes an HTTP endpoint that allows clients to log in by sending
a username and password. The credentials are validated against the
users stored in the database.
```

Definition at line 23 of file AuthController.java.

9.11.2 Member Function Documentation

9.11.2.1 login()

```
ResponseEntity<?> es.ull.esit.server.controller.AuthController.login (
            @RequestBody LoginRequest request)  [inline]
```

Endpoint for handling user login requests:

- receives a username and password in the request body.
- searches for the user in the database.
- compares the provided password with the stored password hash.
- return 200 OK with user data if credentials are valid.
- return 401 Unauthorized if credentials are invalid.

Parameters

request	[LoginRequest] Login request payload containing username and password.

Returns

[ResponseEntity] HTTP response indicating success or failure of authentication.

Definition at line 61 of file AuthController.java.

```
00061                                                                    {
00062
00063     LOG.info("Login attempt for user: {}", request.username);
00064
00065     Optional<User> userOpt = userRepository.findByUsername(request.username);
00066
00067     if (!userOpt.isPresent()) {
00068       LOG.warn("User not found in DB: {}", request.username);
00069       return ResponseEntity.status(401).body("Invalid credentials");
00070     }
00071
00072     User user = userOpt.get();
00073     LOG.debug("User found. Stored hash = {}", user.getPasswordHash()); // Mejor debug, no info
00074
00075     if (!passwordEncoder.matches(request.password, user.getPasswordHash())) {
00076       LOG.warn("Password mismatch for user: {}", request.username);
00077       return ResponseEntity.status(401).body("Invalid credentials");
00078     }
00079
00080     LOG.info("Login OK for user: {}", request.username);
00081
00082     return ResponseEntity.ok(user);
00083
00084   }
```

References LOG, login(), passwordEncoder, and userRepository.

Referenced by login().

Here is the call graph for this function:

Here is the caller graph for this function:

9.11.3 Member Data Documentation

9.11.3.1 LOG

```
final Logger es.ull.esit.server.controller.AuthController.LOG = LoggerFactory.getLogger(Auth↩
Controller.class)  [static], [private]
```

Logger for logging authentication events.

Definition at line 26 of file AuthController.java.

Referenced by login().

9.11.3.2 passwordEncoder

```
PasswordEncoder es.ull.esit.server.controller.AuthController.passwordEncoder  [private]
```

Component used to verify raw passwords against stored hashes.

Definition at line 34 of file AuthController.java.

Referenced by login().

9.11.3.3 userRepository

```
UserRepository es.ull.esit.server.controller.AuthController.userRepository  [private]
```

Repository for accessing user data.

Definition at line 30 of file AuthController.java.

Referenced by login().

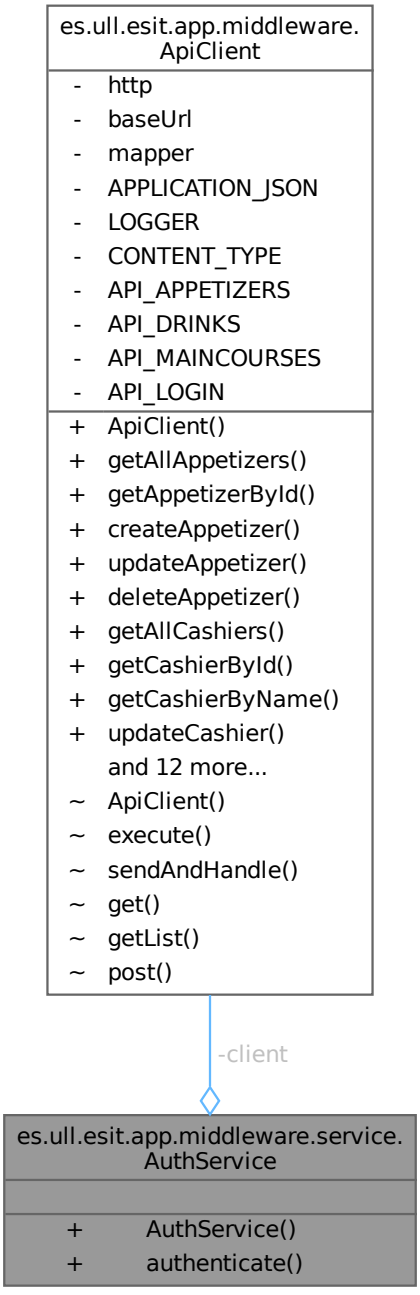
The documentation for this class was generated from the following file:

- AuthController.java

9.12 es.ull.esit.app.middleware.service.AuthService Class Reference

Service responsible for handling authentication logic on the client side.

Collaboration diagram for es.ull.esit.app.middleware.service.AuthService:



Public Member Functions

- AuthService (ApiClient client)

Constructs an AuthService with the given API client.

- User authenticate (String username, String password)

  Attempts to authenticate a user against the backend.

Private Attributes

- final ApiClient client

  REST API client used to communicate with the backend login endpoint.

9.12.1 Detailed Description

Service responsible for handling authentication logic on the client side.

```
Performs local validation of credentials and delegates the actual login
process to the ApiClient.
```

Definition at line 12 of file AuthService.java.

9.12.2 Constructor & Destructor Documentation

9.12.2.1 AuthService()

```
es.ull.esit.app.middleware.service.AuthService.AuthService (
            ApiClient client)  [inline]
```

Constructs an AuthService with the given API client.

Parameters

client	[ApiClient] Client used to perform HTTP requests to the backend.

Definition at line 22 of file AuthService.java.

```
00022                                       {
00023     this.client = client;
00024   }
```

References client.

9.12.3 Member Function Documentation

9.12.3.1 authenticate()

```
User es.ull.esit.app.middleware.service.AuthService.authenticate (
            String username,
            String password)  [inline]
```

Attempts to authenticate a user against the backend.


```
Validates username and password locally, then calls
ApiClient.login(username, password). If validation fails or the
server rejects the credentials, an exception is thrown.
```

Parameters

username	[String] Username entered by the user.
password	[String] Password entered by the user.

Returns

[User] Authenticated user object returned by the backend.

Definition at line 37 of file AuthService.java.

```
00037                                                       {
00038     // 1. Local Validation.
00039     if (username == null || username.trim().isEmpty()) {
00040       throw new IllegalArgumentException("Username cannot be empty.");
00041     }
00042     if (password == null || password.isEmpty()) {
00043       throw new IllegalArgumentException("Password cannot be empty.");
00044     }
00045
00046     // 2. Call API.
00047     return client.login(username, password);
00048   }
```

References client.

9.12.4 Member Data Documentation

9.12.4.1 client

```
final ApiClient es.ull.esit.app.middleware.service.AuthService.client  [private]
```

REST API client used to communicate with the backend login endpoint.

Definition at line 15 of file AuthService.java.

Referenced by authenticate(), and AuthService().

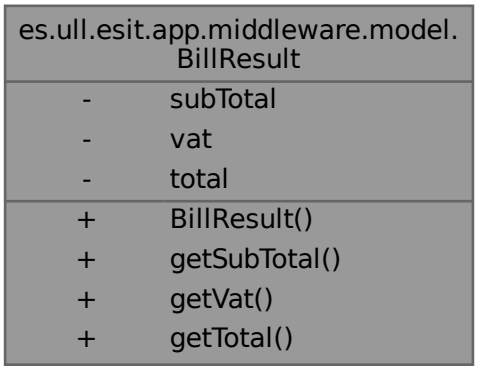
The documentation for this class was generated from the following file:

- AuthService.java

9.13 es.ull.esit.app.middleware.model.BillResult Class Reference

Data Transfer Object representing the result of a bill calculation.

Collaboration diagram for es.ull.esit.app.middleware.model.BillResult:



Public Member Functions

- BillResult (double subTotal, double vat, double total)

  Constructs a bill result with the given amounts.
- double getSubTotal ()

  Gets the subtotal amount.
- double getVat ()

  Gets the VAT amount.
- double getTotal ()

  Gets the total amount.

Private Attributes

- final double subTotal

  Calculated subtotal amount before VAT.
- final double vat

  Calculated VAT amount.
- final double total

  Final total amount including VAT.

9.13.1 Detailed Description

Data Transfer Object representing the result of a bill calculation.

```
It stores subtotal, VAT and total and is used exclusively on the client
side by the OrderService.
```

Definition at line 9 of file BillResult.java.

9.13.2 Constructor & Destructor Documentation

9.13.2.1 BillResult()

```
es.ull.esit.app.middleware.model.BillResult.BillResult (
            double subTotal,
            double vat,
            double total)  [inline]
```

Constructs a bill result with the given amounts.

Parameters

subTotal	[double] Subtotal amount before VAT.
vat	[double] VAT amount.
total	[double] Final total amount including VAT.

Definition at line 27 of file BillResult.java.

```
00027                                                    {
00028     this.subTotal = subTotal;
00029     this.vat = vat;
00030     this.total = total;
00031   }
```

References subTotal, total, and vat.

9.13.3 Member Function Documentation

9.13.3.1 getSubTotal()

```
double es.ull.esit.app.middleware.model.BillResult.getSubTotal ()  [inline]
```

Gets the subtotal amount.

Returns

[double] Subtotal amount before VAT.

Definition at line 38 of file BillResult.java.

```
00038                          {
00039     return subTotal;
00040   }
```

References subTotal.

9.13.3.2 getTotal()

```
double es.ull.esit.app.middleware.model.BillResult.getTotal ()  [inline]
```

Gets the total amount.

Returns

[double] Final total amount including VAT.

Definition at line 56 of file BillResult.java.

```
00056                          {
00057     return total;
00058   }
```

References total.

9.13.3.3 getVat()

```
double es.ull.esit.app.middleware.model.BillResult.getVat ()  [inline]
```

Gets the VAT amount.

Returns

[double] VAT amount.

Definition at line 47 of file BillResult.java.

```
00047                         {
00048     return vat;
00049   }
```

References vat.

9.13.4 Member Data Documentation

9.13.4.1 subTotal

```
final double es.ull.esit.app.middleware.model.BillResult.subTotal  [private]
```

Calculated subtotal amount before VAT.

Definition at line 12 of file BillResult.java.

Referenced by BillResult(), and getSubTotal().

9.13.4.2 total

```
final double es.ull.esit.app.middleware.model.BillResult.total  [private]
```

Final total amount including VAT.

Definition at line 18 of file BillResult.java.

Referenced by BillResult(), and getTotal().

9.13.4.3 vat

```
final double es.ull.esit.app.middleware.model.BillResult.vat  [private]
```

Calculated VAT amount.

Definition at line 15 of file BillResult.java.

Referenced by BillResult(), and getVat().

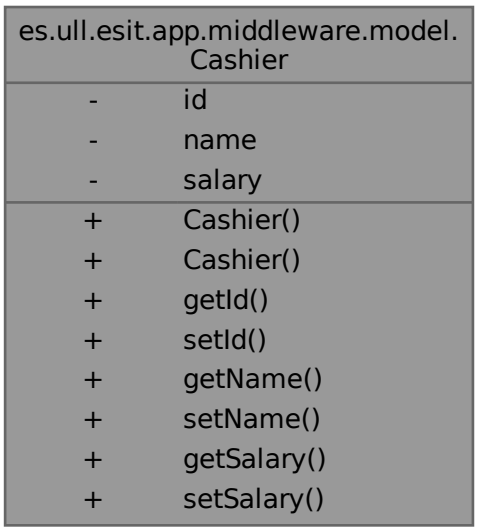
The documentation for this class was generated from the following file:

- BillResult.java

9.14 es.ull.esit.app.middleware.model.Cashier Class Reference

Client-side model representing a cashier returned by the backend.

Collaboration diagram for es.ull.esit.app.middleware.model.Cashier:



Public Member Functions

- Cashier ()

  Default constructor required for JSON deserialization.
- Cashier (Long id, String name, Integer salary)

  Constructs a cashier with all fields.
- Long getId ()

  Gets the cashier identifier.
- void setId (Long id)

  Sets the cashier identifier.
- String getName ()

  Gets the cashier name.
- void setName (String name)

  Sets the cashier name.
- Integer getSalary ()

  Gets the cashier salary.
- void setSalary (Integer salary)

  Sets the cashier salary.

Private Attributes

- Long id

  Unique identifier of the cashier (JSON property "id").
- String name

  Cashier name (JSON property "name").
- Integer salary

  Cashier salary (JSON property "salary").

9.14.1 Detailed Description

Client-side model representing a cashier returned by the backend.

```
It is used by the Swing application to display and manage cashier
information obtained from the "/api/cashiers" endpoint.
```

Definition at line 11 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

9.14.2 Constructor & Destructor Documentation

9.14.2.1 Cashier() [1/2]

```
es.ull.esit.app.middleware.model.Cashier.Cashier ()  [inline]
```

Default constructor required for JSON deserialization.

Definition at line 28 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00028                     {
00029     }
```

References Cashier().

Referenced by Cashier().

Here is the call graph for this function:

Here is the caller graph for this function:

9.14.2.2 Cashier() [2/2]

```
es.ull.esit.app.middleware.model.Cashier.Cashier (
            Long id,
            String name,
            Integer salary)  [inline]
```

Constructs a cashier with all fields.

Parameters

id	[Long] Unique identifier of the cashier.
name	[String] Name of the cashier.
salary	[Integer] Salary of the cashier.

Definition at line 38 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00038                                                   {
00039     this.id = id;
00040     this.name = name;
00041     this.salary = salary;
00042   }
```

References id, name, and salary.

9.14.3 Member Function Documentation

9.14.3.1 getId()

```
Long es.ull.esit.app.middleware.model.Cashier.getId ()  [inline]
```

Gets the cashier identifier.

Returns

[Long] Unique identifier of the cashier.

Definition at line 49 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00049                       {
00050     return id;
00051   }
```

References id.

9.14.3.2 getName()

```
String es.ull.esit.app.middleware.model.Cashier.getName ()  [inline]
```

Gets the cashier name.

Returns

[String] Name of the cashier.

Definition at line 67 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00067                           {
00068     return name;
00069   }
```

References name.

9.14.3.3 getSalary()

```
Integer es.ull.esit.app.middleware.model.Cashier.getSalary ()  [inline]
```

Gets the cashier salary.

Returns

[Integer] Salary of the cashier.

Definition at line 85 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00085                             {
00086     return salary;
00087   }
```

References salary.

9.14.3.4 setId()

```
void es.ull.esit.app.middleware.model.Cashier.setId (
            Long id)  [inline]
```

Sets the cashier identifier.

Parameters

id	[Long] Unique identifier of the cashier.

Definition at line 58 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00058                             {
00059     this.id = id;
00060   }
```

References id.

9.14.3.5 setName()

```
void es.ull.esit.app.middleware.model.Cashier.setName (
```

Parameters

name	[String] Name of the cashier.

Definition at line 76 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00076                                   {
00077     this.name = name;
00078   }
```

References name.

9.14.3.6 setSalary()

```
void es.ull.esit.app.middleware.model.Cashier.setSalary (
             Integer salary)  [inline]
```

Sets the cashier salary.

Parameters

salary	[Integer] Salary of the cashier.

Definition at line 94 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

```
00094                                      {
00095     this.salary = salary;
00096   }
```

References salary.

9.14.4 Member Data Documentation

9.14.4.1 id

```
Long es.ull.esit.app.middleware.model.Cashier.id  [private]
```

Unique identifier of the cashier (JSON property "id").

Definition at line 15 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

Referenced by Cashier(), getId(), and setId().

9.14.4.2 name

```
String es.ull.esit.app.middleware.model.Cashier.name  [private]
```

Cashier name (JSON property "name").

Definition at line 19 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

Referenced by Cashier(), getName(), and setName().

9.14.4.3 salary

```
Integer es.ull.esit.app.middleware.model.Cashier.salary  [private]
```

Cashier salary (JSON property "salary").

Definition at line 23 of file src/main/java/es/ull/esit/app/middleware/model/Cashier.java.

Referenced by Cashier(), getSalary(), and setSalary().

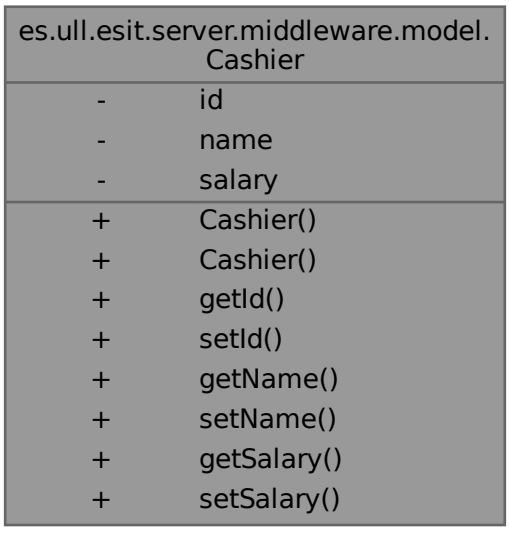
The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/app/middleware/model/Cashier.java

9.15 es.ull.esit.server.middleware.model.Cashier Class Reference

JPA entity that represents a cashier in the system.

Collaboration diagram for es.ull.esit.server.middleware.model.Cashier:



Public Member Functions

- Cashier ()

  Default constructor required by JPA.
- Cashier (Long id, String name, Integer salary)

  Full constructor.
- Long getId ()

  Gets the cashier identifier.

- void setId (Long id)

  Sets the cashier identifier.
- String getName ()

  Gets the cashier name (username).
- void setName (String name)

  Sets the cashier name (username).
- Integer getSalary ()

  Gets the cashier salary.
- void setSalary (Integer salary)

  Sets the cashier salary.

Private Attributes

- Long id

  Primary key (unique identifier) of the cashier (column "cashier_id").
- String name

  Cashier username (column "cashier_name").
- Integer salary

  Cashier salary (column "cashier_salary").

9.15.1 Detailed Description

JPA entity that represents a cashier in the system.

```
It is mapped to the "cashier" table used by the REST API.
Each row contains a unique identifier, name (username), and salary.
```

Definition at line 15 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

9.15.2 Constructor & Destructor Documentation

9.15.2.1 Cashier() [1/2]

```
es.ull.esit.server.middleware.model.Cashier.Cashier ()  [inline]
```

Default constructor required by JPA.

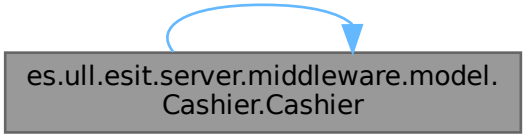
Definition at line 33 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00033                     {
00034     }
```

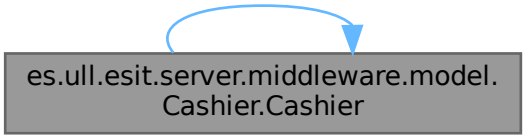
References Cashier().

Referenced by Cashier().

Here is the call graph for this function:



Here is the caller graph for this function:



9.15.2.2 Cashier() [2/2]

```
es.ull.esit.server.middleware.model.Cashier.Cashier (
            Long id,
            String name,
            Integer salary)  [inline]
```

Full constructor.

Parameters

id	[Long] Identifier of the cashier.
name	[String] Username of the cashier.
salary	[Integer] Salary of the cashier.

Definition at line 43 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00043                                                   {
00044     this.id = id;
00045     this.name = name;
00046     this.salary = salary;
00047   }
```

References id, name, and salary.

9.15.3 Member Function Documentation

9.15.3.1 getId()

```
Long es.ull.esit.server.middleware.model.Cashier.getId ()  [inline]
```

Gets the cashier identifier.

Returns

[Long] Cashier id.

Definition at line 54 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00054                       {
00055     return id;
00056   }
```

References id.

9.15.3.2 getName()

```
String es.ull.esit.server.middleware.model.Cashier.getName ()  [inline]
```

Gets the cashier name (username).

Returns

[String] Cashier name (username).

Definition at line 73 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00073                           {
00074     return name;
00075   }
```

References name.

9.15.3.3 getSalary()

```
Integer es.ull.esit.server.middleware.model.Cashier.getSalary ()  [inline]
```

Gets the cashier salary.

Returns

[Integer] cashier salary.

Definition at line 91 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00091                             {
00092     return salary;
00093   }
```

References salary.

9.15.3.4 setId()

Parameters

id	[Long] New cashier id.

Definition at line 64 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00064                                {
00065     this.id = id;
00066   }
```

References id.

9.15.3.5 setName()

```
void es.ull.esit.server.middleware.model.Cashier.setName (
             String name)  [inline]
```

Sets the cashier name (username).

Parameters

name	[String] New cashier name (username).

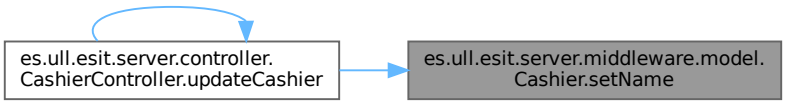
Definition at line 82 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00082                                  {
00083     this.name = name;
00084   }
```

References name.

Referenced by es.ull.esit.server.controller.CashierController.updateCashier().

Here is the caller graph for this function:



9.15.3.6 setSalary()

```
void es.ull.esit.server.middleware.model.Cashier.setSalary (
             Integer salary)  [inline]
```

Sets the cashier salary.

Parameters

salary	[Integer] New cashier salary.

Definition at line 100 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

```
00100                                       {
00101     this.salary = salary;
00102   }
```

References salary.

9.15.4 Member Data Documentation

9.15.4.1 id

```
Long es.ull.esit.server.middleware.model.Cashier.id  [private]
```

Primary key (unique identifier) of the cashier (column "cashier_id").

Definition at line 20 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

Referenced by Cashier(), getId(), and setId().

9.15.4.2 name

```
String es.ull.esit.server.middleware.model.Cashier.name  [private]
```

Cashier username (column "cashier_name").

Definition at line 24 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

Referenced by Cashier(), getName(), and setName().

9.15.4.3 salary

```
Integer es.ull.esit.server.middleware.model.Cashier.salary  [private]
```

Cashier salary (column "cashier_salary").

Definition at line 28 of file server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java.

Referenced by Cashier(), getSalary(), and setSalary().

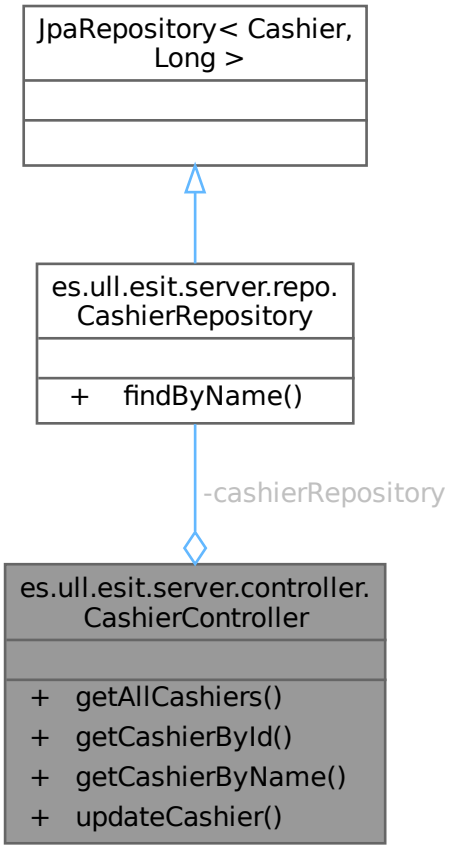
The documentation for this class was generated from the following file:

- server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java

9.16 es.ull.esit.server.controller.CashierController Class Reference

REST controller for managing cashiers.

Collaboration diagram for es.ull.esit.server.controller.CashierController:



Public Member Functions

- ResponseEntity< List< Cashier > > getAllCashiers ()
  Returns all cashiers.
- ResponseEntity< Cashier > getCashierById (@PathVariable Long id)
  Returns a single cashier by ID.
- ResponseEntity< Cashier > getCashierByName (@PathVariable String name)
  Returns a single cashier by name (username).
- ResponseEntity< Cashier > updateCashier (@PathVariable Long id, @RequestBody Cashier cashier)
  Updates an existing cashier, but only the name and salary fields.

Private Attributes

- CashierRepository cashierRepository

9.16.1 Detailed Description

REST controller for managing cashiers.

```
Provides CRUD operations for Cashier entities using the path
"/api/cashiers".

Cashiers are created automatically when users with 'CASHIER' role are
added to the table 'users'.
This controller is mainly READ-ONLY and allows updating salary/name.
```

Definition at line 25 of file CashierController.java.

9.16.2 Member Function Documentation

9.16.2.1 getAllCashiers()

```
ResponseEntity< List< Cashier > > es.ull.esit.server.controller.CashierController.getAll↩
Cashiers ()  [inline]
```

Returns all cashiers.

```
HTTP GET /api/cashiers
```

Returns

[ResponseEntity<List<Cashier>>] List of cashiers with HTTP 200 status.

Definition at line 40 of file CashierController.java.

```
00040                                                    {
00041     List<Cashier> cashiers = cashierRepository.findAll();
00042     return ResponseEntity.ok(cashiers);
00043   }
```

References cashierRepository.

9.16.2.2 getCashierById()

```
ResponseEntity< Cashier > es.ull.esit.server.controller.CashierController.getCashierById (
            @PathVariable Long id)  [inline]
```

Returns a single cashier by ID.

```
HTTP GET /api/cashiers/{id}
```

Parameters

id	[Long] Identifier of the cashier.

Returns

[ResponseEntity<Cashier>] The cashier if found with HTTP 200 status, or HTTP 404 if not found.

Definition at line 55 of file CashierController.java.

```
00055                                                                    {
00056     Optional<Cashier> cashier = cashierRepository.findById(id);
00057     return cashier
00058         .map(ResponseEntity::ok)
00059         .orElseGet(() -> ResponseEntity.notFound().build());
00060   }
```

References cashierRepository, and getCashierById().

Referenced by getCashierById().

Here is the call graph for this function:

Here is the caller graph for this function:

9.16.2.3 getCashierByName()

```
ResponseEntity< Cashier > es.ull.esit.server.controller.CashierController.getCashierByName (
            @PathVariable String name)  [inline]
```

Returns a single cashier by name (username).

```
HTTP GET /api/cashiers/name/{name}
```

Parameters

name	[String] Name of the cashier.

Returns

[ResponseEntity<Cashier>] The cashier if found with HTTP 200 status, or HTTP 404 if not found.

Definition at line 72 of file CashierController.java.

```
00072                                                                    {
00073     Optional<Cashier> cashier = cashierRepository.findByName(name);
00074     return cashier
00075         .map(ResponseEntity::ok)
00076         .orElseGet(() -> ResponseEntity.notFound().build());
00077   }
```

References cashierRepository, and getCashierByName().

Referenced by getCashierByName().

Here is the call graph for this function:

Here is the caller graph for this function:

9.16.2.4 updateCashier()

```
ResponseEntity< Cashier > es.ull.esit.server.controller.CashierController.updateCashier (
            @PathVariable Long id,
            @RequestBody Cashier cashier)  [inline]
```


Updates an existing cashier, but only the name and salary fields.

```
HTTP PUT /api/cashiers/{id}
```

Parameters

id	[Long] Identifier of the cashier to update.
cashier	[Cashier] New data for the cashier.

Returns

[ResponseEntity<Cashier>] The updated cashier with HTTP 200 status, or HTTP 404 if not found.

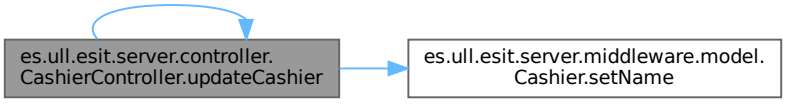
Definition at line 90 of file CashierController.java.

```
00091                                     {
00092     Optional<Cashier> existingOpt = cashierRepository.findById(id);
00093     if (!existingOpt.isPresent()) {
00094       return ResponseEntity.notFound().build();
00095     }
00096
00097     Cashier existing = existingOpt.get();
00098     existing.setName(cashier.getName());
00099     existing.setSalary(cashier.getSalary());
00100
00101     Cashier updated = cashierRepository.save(existing);
00102     return ResponseEntity.ok(updated);
00103   }
```

References cashierRepository, es.ull.esit.server.middleware.model.Cashier.setName(), and updateCashier().

Referenced by updateCashier().

Here is the call graph for this function:



Here is the caller graph for this function:

9.16.3 Member Data Documentation

9.16.3.1 cashierRepository

```
CashierRepository es.ull.esit.server.controller.CashierController.cashierRepository  [private]
```

Definition at line 29 of file CashierController.java.

Referenced by getAllCashiers(), getCashierById(), getCashierByName(), and updateCashier().

The documentation for this class was generated from the following file:

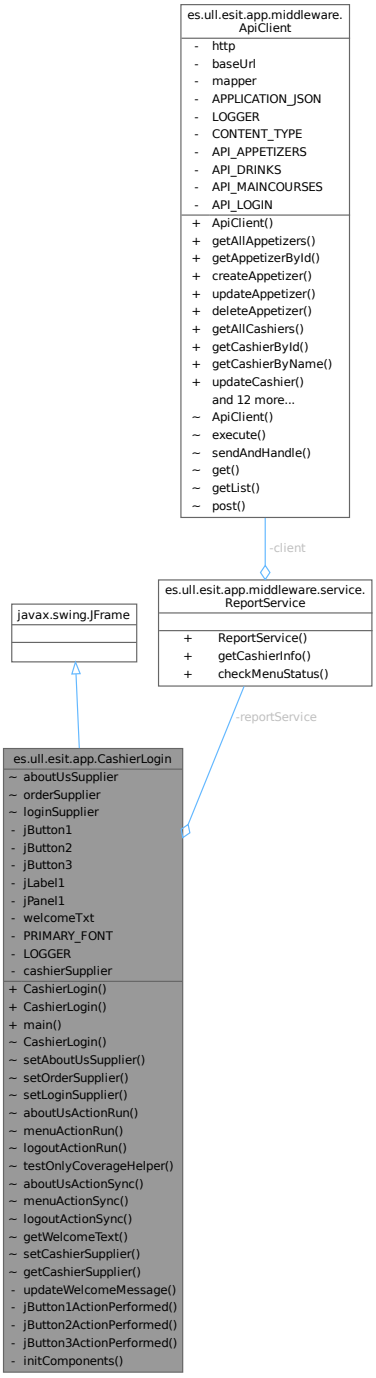
- CashierController.java

9.17 es.ull.esit.app.CashierLogin Class Reference

Login window for authenticating cashiers.

Inheritance diagram for es.ull.esit.app.CashierLogin:

Collaboration diagram for es.ull.esit.app.CashierLogin:



Public Member Functions

- CashierLogin ()

  Default constructor.
- CashierLogin (String name)

  Constructor with cashier name.

Static Public Member Functions

- static void main (String[ ] args)

  Standalone entry point for testing the Cashier window.

Private Member Functions

- void updateWelcomeMessage (String name)

  Updates the welcome message label.
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the "About us" button.
- void jButton2ActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the "Menu" button.
- void jButton3ActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the "LogOut" button.
- void initComponents ()

  Initializes GUI components.

Private Attributes

- final transient ReportService reportService

  Service used to retrieve reports and basic status from the backend.
- javax.swing.JButton jButton1

  Button that opens the "About us" information window.
- javax.swing.JButton jButton2

  Button that opens the general menu window (Frame1).
- javax.swing.JButton jButton3

  Button used to log out and return to the login screen.
- javax.swing.JLabel jLabel1

  Label that displays the logo or application image.
- javax.swing.JPanel jPanel1

  Main container panel for all UI elements.
- javax.swing.JLabel welcomeTxt

  Label that displays the welcome message for the cashier.

Static Private Attributes

- static final String PRIMARY_FONT = "Yu Gothic UI"
- static final Logger LOGGER = LoggerFactory.getLogger(CashierLogin.class)

  Logger instance for logging events and errors.
- static Supplier<? extends javax.swing.JFrame > cashierSupplier = () -> new CashierLogin()

  Supplier for the top-level cashier window used by main(); tests can override it.

9.17.1 Detailed Description

Login window for authenticating cashiers.

```
It is displayed after a successful login with role CASHIER.
Shows a Swing form that lets cashiers:
- access to the general menu window used to register orders.
- access to the "About us" information window.
- log out and return to the login window.

Additionally, some actions perform background calls to the backend
using ReportService to:
- retrieve cashier information.
- check the menu or system status.
```

Definition at line 27 of file CashierLogin.java.

9.17.2 Constructor & Destructor Documentation

9.17.2.1 CashierLogin() [1/2]

```
es.ull.esit.app.CashierLogin.CashierLogin ()  [inline]
```

Default constructor.

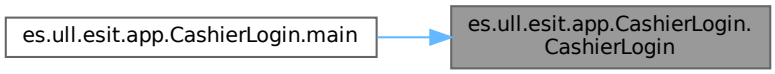
```
Creates a cashier window without a specific name in the welcome
message.
Internally delegates to the constructor that accepts a name.
```

Definition at line 44 of file CashierLogin.java.

```
00044                       {
00045     this((String) null);
00046   }
```

Referenced by main().

Here is the caller graph for this function:



9.17.2.2 CashierLogin() [2/2]

```
es.ull.esit.app.CashierLogin.CashierLogin (
            String name)  [inline]
```

Constructor with cashier name.


```
Creates the cashier window, initializes the GUI, builds an
ApiClient pointing to the backend, creates a ReportService and
updates the welcome message using the cashier name.
```

Parameters

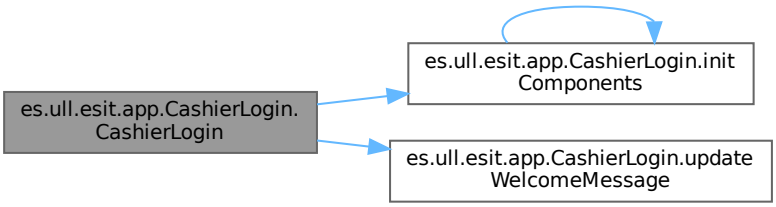
name	[String] Name of the cashier returned by the backend (or null for a generic welcome message).

Definition at line 58 of file CashierLogin.java.

```
00058                                     {
00059     initComponents();
00060
00061     ApiClient client = new ApiClient("http://localhost:8080");
00062     this.reportService = new ReportService(client);
00063
00064     updateWelcomeMessage(name);
00065   }
```

References initComponents(), and updateWelcomeMessage().

Here is the call graph for this function:



9.17.3 Member Function Documentation

9.17.3.1 initComponents()

```
void es.ull.esit.app.CashierLogin.initComponents ()  [inline], [private]
```

Initializes GUI components.

```
Generated by the Form Editor: do not modify.
Sets up the welcome label, logo image, buttons for "About us", "Menu",
"LogOut",
and the layout and configuration of the window.
```

Definition at line 372 of file CashierLogin.java.

```
00372                                     {
00373
00374     jPanel1 = new javax.swing.JPanel();
00375     welcomeTxt = new javax.swing.JLabel();
00376     jButton1 = new javax.swing.JButton();
00377     jButton2 = new javax.swing.JButton();
00378     jLabel1 = new javax.swing.JLabel();
00379     jButton3 = new javax.swing.JButton();
00380
00381     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00382     setTitle("Cashier");
00383     setResizable(false);
00384
00385     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00386
```

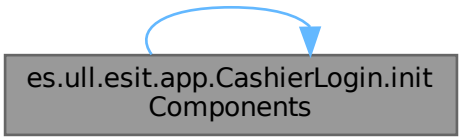
```
00387     welcomeTxt.setFont(new java.awt.Font(PRIMARY_FONT, 1, 24)); // NOI18N
00388     welcomeTxt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00389     welcomeTxt.setText("Welcome Cashier");
00390
00391     jButton1.setBackground(new java.awt.Color(153, 153, 153));
00392     jButton1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00393     jButton1.setText("About us");
00394     jButton1.addActionListener(this::jButton1ActionPerformed);
00395
00396     jButton2.setBackground(new java.awt.Color(153, 153, 153));
00397     jButton2.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00398     jButton2.setText("Menu");
00399     jButton2.addActionListener(this::jButton2ActionPerformed);
00400
00401     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
      NOI18N
00402
00403     jButton3.setBackground(new java.awt.Color(255, 255, 255));
00404     jButton3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00405     jButton3.setText("LogOut");
00406     jButton3.addActionListener(this::jButton3ActionPerformed);
00407
00408     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00409     jPanel1.setLayout(jPanel1Layout);
00410     jPanel1Layout.setHorizontalGroup(
00411         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00412             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00413                 .addContainerGap(109, Short.MAX_VALUE)
00414                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00415                     .addComponent(welcomeTxt, javax.swing.GroupLayout.PREFERRED_SIZE, 299,
00416                         javax.swing.GroupLayout.PREFERRED_SIZE)
00417
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00418                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00419                             .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00420                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00421                             .addGap(98, 98, 98))
00422                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00423                             .addComponent(jLabel1)
00424                             .addGap(183, 183, 183))
00425                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00426                             .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00427                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00428                             .addGap(190, 190, 190)))))
00429             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00430                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00431                     .addContainerGap(108, Short.MAX_VALUE)
00432                     .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00433                         javax.swing.GroupLayout.PREFERRED_SIZE)
00434                     .addGap(99, 99, 99))));
00435     jPanel1Layout.setVerticalGroup(
00436         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00437             .addGroup(jPanel1Layout.createSequentialGroup()
00438                 .addGap(45, 45, 45)
00439                 .addComponent(welcomeTxt, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
00440                     javax.swing.GroupLayout.PREFERRED_SIZE)
00441                 .addGap(18, 18, 18)
00442                 .addComponent(jLabel1)
00443                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 99,
      Short.MAX_VALUE)
00444                 .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00445                     javax.swing.GroupLayout.PREFERRED_SIZE)
00446                 .addGap(57, 57, 57)
00447                 .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00448                     javax.swing.GroupLayout.PREFERRED_SIZE)
00449                 .addGap(68, 68, 68))
00450             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00451                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00452                     .addContainerGap(290, Short.MAX_VALUE)
00453                     .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00454                         javax.swing.GroupLayout.PREFERRED_SIZE)
00455                     .addGap(242, 242, 242))));
00456
00457     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00458     getContentPane().setLayout(layout);
00459     layout.setHorizontalGroup(
00460         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00461             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00462                 Short.MAX_VALUE));
00463     layout.setVerticalGroup(
```

```
00464         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00465             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00466                 Short.MAX_VALUE));
00467
00468     pack();
00469     setLocationRelativeTo(null);
00470   }// </editor-fold>//GEN-END:initComponents
```

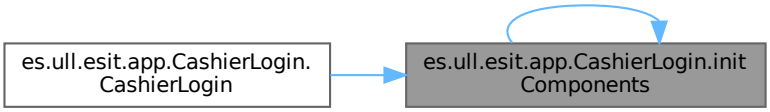
References initComponents(), jButton1, jButton2, jButton3, jLabel1, jPanel1, PRIMARY_FONT, and welcomeTxt.

Referenced by CashierLogin(), and initComponents().

Here is the call graph for this function:



Here is the caller graph for this function:



9.17.3.2 jButton1ActionPerformed()

```
void es.ull.esit.app.CashierLogin.jButton1ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "About us" button.

```
   Action steps:
   - Starts a background thread.
   - Uses ReportService.getCashierInfo() to retrieve cashier data from
   the backend and logs the number of cashiers found.
   - Opens the Info window in the Event Dispatch Thread.
   - Closes the current Cashier window.

   Any error retrieving data is logged to the standard error stream
   but does not prevent opening the Info window.
```

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by button click.

Definition at line 225 of file CashierLogin.java.

```
00225                                                                    {
00226     new Thread(() -> {
00227       try {
00228         List<es.ull.esit.app.middleware.model.Cashier> cashiers = reportService.getCashierInfo();
00229         LOGGER.info("Found {} cashiers in DB.", cashiers.size());
00230
00231       } catch (Exception ex) {
00232         LOGGER.warn("Warning: Could not fetch cashier stats: {}", ex.getMessage());
00233       }
00234       SwingUtilities.invokeLater(() -> {
00235         aboutUsSupplier.get().setVisible(true);
00236         dispose();
00237       });
00238     }).start();
00239   }
```

References LOGGER, and reportService.

9.17.3.3 jButton2ActionPerformed()

```
void es.ull.esit.app.CashierLogin.jButton2ActionPerformed (
             java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Menu" button.

```
Action steps:
- Starts a background thread.
- Uses ReportService.checkMenuStatus() to verify communication
with the backend and logs the returned status.
- Opens the general menu window (Frame1) in the Event Dispatch
Thread.
- Closes the current Cashier window.

Any error while checking the status is logged to the standard
error stream but does not prevent opening the menu window.
```

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by button click.

Definition at line 272 of file CashierLogin.java.

```
00272                                                                    {
00273     new Thread(() -> {
00274       try {
00275         String status = reportService.checkMenuStatus();
00276         LOGGER.info("Menu status: {}", status);
00277
00278       } catch (Exception ex) {
00279         LOGGER.error("Error checking menu status: {}", ex.getMessage());
00280       }
00281
00282       SwingUtilities.invokeLater(() -> {
00283         orderSupplier.get().setVisible(true);
00284         dispose();
00285       });
00286     }).start();
00287   }
```

References LOGGER, and reportService.

9.17.3.4 jButton3ActionPerformed()

```
void es.ull.esit.app.CashierLogin.jButton3ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "LogOut" button.

```
Action steps:
- Opens the Login window.
- Closes the current Cashier window.

Any server-side logout or token invalidation, if needed, should be
handled outside this class.
```

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by button click.

Definition at line 315 of file CashierLogin.java.

```
00315                                                                    {
00316     loginSupplier.get().setVisible(true);
00317     dispose();
00318   }
```

9.17.3.5 main()

```
void es.ull.esit.app.CashierLogin.main (
            String[] args)  [inline], [static]
```

Standalone entry point for testing the Cashier window.

```
Sets the Nimbus look and feel if available and shows an instance
of Cashier without a specific cashier name. In the normal
application flow this window is started from the Login class after
a successful cashier login.
```

Parameters

args	[String[]] Command line arguments (not used).

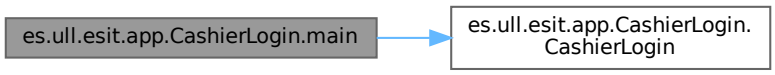
Definition at line 345 of file CashierLogin.java.

```
00345                                       {
00346     try {
00347       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00348         if ("Nimbus".equals(info.getName())) {
00349           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00350           break;
00351         }
00352       }
00353     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00354         | javax.swing.UnsupportedLookAndFeelException ex) {
00355
      java.util.logging.Logger.getLogger(CashierLogin.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00356     }
00357
00358     java.awt.EventQueue.invokeLater(() -> getCashierSupplier().get().setVisible(true));
```

```
00359   }
```

References CashierLogin().

Here is the call graph for this function:



9.17.3.6 updateWelcomeMessage()

```
void es.ull.esit.app.CashierLogin.updateWelcomeMessage (
            String name)  [inline], [private]
```

Updates the welcome message label.

```
If the name is null or empty, the label shows "Welcome Cashier".
Otherwise the label shows "Welcome " followed by the given name.
```

Parameters

name	[String] Cashier name or null.

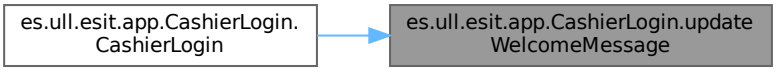
Definition at line 201 of file CashierLogin.java.

```
00201                                                {
00202     if (name == null || name.trim().isEmpty()) {
00203       welcomeTxt.setText("Welcome Cashier");
00204     } else {
00205       welcomeTxt.setText("Welcome " + name);
00206     }
00207   }
```

References welcomeTxt.

Referenced by CashierLogin().

Here is the caller graph for this function:



9.17.4 Member Data Documentation

9.17.4.1 cashierSupplier

```
Supplier<? extends javax.swing.JFrame> es.ull.esit.app.CashierLogin.cashierSupplier = () ->
new CashierLogin() [static], [private]
```

Supplier for the top-level cashier window used by main(); tests can override it.

Definition at line 97 of file CashierLogin.java.

9.17.4.2 jButton1

```
javax.swing.JButton es.ull.esit.app.CashierLogin.jButton1 [private]
```

Button that opens the "About us" information window.

Definition at line 476 of file CashierLogin.java.

Referenced by initComponents().

9.17.4.3 jButton2

```
javax.swing.JButton es.ull.esit.app.CashierLogin.jButton2 [private]
```

Button that opens the general menu window (Frame1).

Definition at line 478 of file CashierLogin.java.

Referenced by initComponents().

9.17.4.4 jButton3

```
javax.swing.JButton es.ull.esit.app.CashierLogin.jButton3 [private]
```

Button used to log out and return to the login screen.

Definition at line 480 of file CashierLogin.java.

Referenced by initComponents().

9.17.4.5 jLabel1

```
javax.swing.JLabel es.ull.esit.app.CashierLogin.jLabel1 [private]
```

Label that displays the logo or application image.

Definition at line 482 of file CashierLogin.java.

Referenced by initComponents().

9.17.4.6 jPanel1

```
javax.swing.JPanel es.ull.esit.app.CashierLogin.jPanel1  [private]
```

Main container panel for all UI elements.

Definition at line 484 of file CashierLogin.java.

Referenced by initComponents().

9.17.4.7 LOGGER

```
final Logger es.ull.esit.app.CashierLogin.LOGGER = LoggerFactory.getLogger(CashierLogin.class)
[static], [private]
```

Logger instance for logging events and errors.

Definition at line 35 of file CashierLogin.java.

Referenced by jButton1ActionPerformed(), and jButton2ActionPerformed().

9.17.4.8 PRIMARY_FONT

```
final String es.ull.esit.app.CashierLogin.PRIMARY_FONT = "Yu Gothic UI"  [static], [private]
```

Definition at line 32 of file CashierLogin.java.

Referenced by initComponents().

9.17.4.9 reportService

```
final transient ReportService es.ull.esit.app.CashierLogin.reportService  [private]
```

Service used to retrieve reports and basic status from the backend.

Definition at line 30 of file CashierLogin.java.

Referenced by jButton1ActionPerformed(), and jButton2ActionPerformed().

9.17.4.10 welcomeTxt

```
javax.swing.JLabel es.ull.esit.app.CashierLogin.welcomeTxt  [private]
```

Label that displays the welcome message for the cashier.

Definition at line 486 of file CashierLogin.java.

Referenced by initComponents(), and updateWelcomeMessage().

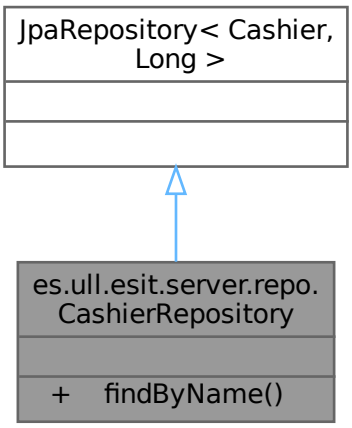
The documentation for this class was generated from the following file:

- CashierLogin.java

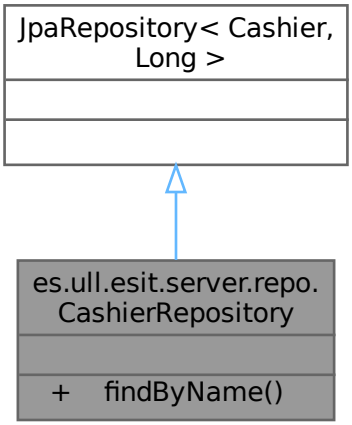
9.18 es.ull.esit.server.repo.CashierRepository Interface Reference

Repository interface for managing cashiers in the database.

Inheritance diagram for es.ull.esit.server.repo.CashierRepository:



Collaboration diagram for es.ull.esit.server.repo.CashierRepository:



Public Member Functions

- Optional< Cashier > findByName (String name)

  Finds a cashier by their name.

9.18.1 Detailed Description

Repository interface for managing cashiers in the database.

```
Extends JpaRepository to provide basic CRUD operations on the
"Cashier" table, such as findAll, findById, save and delete.
```

Definition at line 14 of file CashierRepository.java.

9.18.2 Member Function Documentation

9.18.2.1 findByName()

```
Optional< Cashier > es.ull.esit.server.repo.CashierRepository.findByName (
            String name)
```

Finds a cashier by their name.

Parameters

name	[String] Name of the cashier.

Returns

[Optional<Cashier>] The cashier if found, or empty if not found.

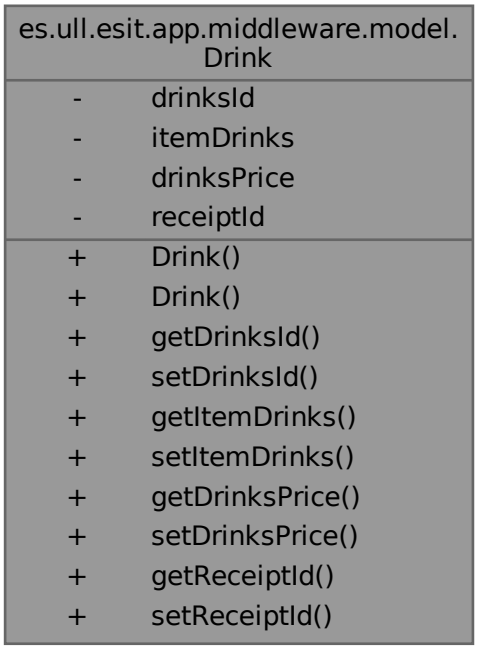
The documentation for this interface was generated from the following file:

- CashierRepository.java

9.19 es.ull.esit.app.middleware.model.Drink Class Reference

Client-side model representing a drink returned by the backend API.

Collaboration diagram for es.ull.esit.app.middleware.model.Drink:



Public Member Functions

- Drink ()

  Default constructor required for JSON deserialization.
- Drink (Long drinksId, String itemDrinks, Integer drinksPrice, Long receiptId)

  Constructs a drink with all fields.
- Long getDrinksId ()

  Gets the drink identifier.
- void setDrinksId (Long drinksId)

  Sets the drink identifier.
- String getItemDrinks ()

  Gets the drink item name.
- void setItemDrinks (String itemDrinks)

  Sets the drink item name.
- Integer getDrinksPrice ()

  Gets the drink price.
- void setDrinksPrice (Integer drinksPrice)

  Sets the drink price.
- Long getReceiptId ()

  Gets the identifier of the related receipt.
- void setReceiptId (Long receiptId)

  Sets the identifier of the related receipt.

Private Attributes

- Long drinksId

  Unique identifier of the drink (JSON property "drinksId").
- String itemDrinks

  Name of the drink item (JSON property "itemDrinks").
- Integer drinksPrice

  Price of the drink (JSON property "drinksPrice").
- Long receiptId

  Identifier of the receipt this drink belongs to (JSON property "receiptId").

9.19.1 Detailed Description

Client-side model representing a drink returned by the backend API.

```
This class is used to deserialize and manipulate drink data obtained
from the "/api/drinks" endpoint.
```

Definition at line 11 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

9.19.2 Constructor & Destructor Documentation

9.19.2.1 Drink() [1/2]

```
es.ull.esit.app.middleware.model.Drink.Drink ()  [inline]
```

Default constructor required for JSON deserialization.

Definition at line 35 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00035                 {
00036     }
```

References Drink().

Referenced by Drink().

Here is the call graph for this function:

Here is the caller graph for this function:

9.19.2.2 Drink() [2/2]

```
es.ull.esit.app.middleware.model.Drink.Drink (
            Long drinksId,
            String itemDrinks,
            Integer drinksPrice,
            Long receiptId)  [inline]
```

Constructs a drink with all fields.

Parameters

drinksId	[Long] Unique identifier of the drink.
itemDrinks	[String] Name of the drink item.
drinksPrice	[Integer] Price of the drink.
receiptId	[Long] Identifier of the related receipt (optional).

Definition at line 46 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00046                                                                                {
00047         this.drinksId = drinksId;
00048         this.itemDrinks = itemDrinks;
00049         this.drinksPrice = drinksPrice;
00050         this.receiptId = receiptId;
00051     }
```

References drinksId, drinksPrice, itemDrinks, and receiptId.

9.19.3 Member Function Documentation

9.19.3.1 getDrinksId()

```
Long es.ull.esit.app.middleware.model.Drink.getDrinksId ()  [inline]
```

Gets the drink identifier.

Returns

[Long] Unique identifier of the drink.

Definition at line 58 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00058                             {
00059     return drinksId;
00060   }
```

References drinksId.

9.19.3.2 getDrinksPrice()

```
Integer es.ull.esit.app.middleware.model.Drink.getDrinksPrice ()  [inline]
```

Gets the drink price.

Returns

[Integer] Price of the drink.

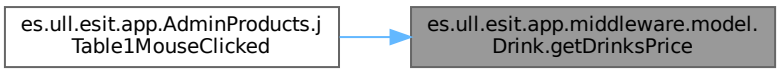
Definition at line 94 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00094                                 {
00095     return drinksPrice;
00096   }
```

References drinksPrice.

Referenced by es.ull.esit.app.AdminProducts.jTable1MouseClicked().

Here is the caller graph for this function:



9.19.3.3 getItemDrinks()

```
String es.ull.esit.app.middleware.model.Drink.getItemDrinks ()  [inline]
```

Gets the drink item name.

Returns

[String] Name of the drink item.

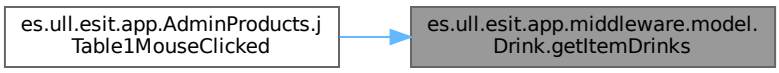
Definition at line 76 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00076                                {
00077     return itemDrinks;
00078   }
```

References itemDrinks.

Referenced by es.ull.esit.app.AdminProducts.jTable1MouseClicked().

Here is the caller graph for this function:



9.19.3.4 getReceiptId()

```
Long es.ull.esit.app.middleware.model.Drink.getReceiptId ()  [inline]
```

Gets the identifier of the related receipt.

Returns

[Long] Identifier of the receipt or null if not set.

Definition at line 112 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00112                               {
00113     return receiptId;
00114   }
```

References receiptId.

9.19.3.5 setDrinksId()

```
void es.ull.esit.app.middleware.model.Drink.setDrinksId (
            Long drinksId)  [inline]
```

Sets the drink identifier.

Parameters

drinksId	[Long] Unique identifier of the drink.

Definition at line 67 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00067                                    {
00068     this.drinksId = drinksId;
00069   }
```

References drinksId.

9.19.3.6 setDrinksPrice()

```
void es.ull.esit.app.middleware.model.Drink.setDrinksPrice (
            Integer drinksPrice)  [inline]
```

Sets the drink price.

Parameters

drinksPrice	[Integer] Price of the drink.

Definition at line 103 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00103                                                {
00104     this.drinksPrice = drinksPrice;
00105   }
```

References drinksPrice.

9.19.3.7 setItemDrinks()

```
void es.ull.esit.app.middleware.model.Drink.setItemDrinks (
            String itemDrinks)  [inline]
```

Sets the drink item name.

Parameters

itemDrinks	[String] Name of the drink item.

Definition at line 85 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00085                                               {
00086     this.itemDrinks = itemDrinks;
00087   }
```

References itemDrinks.

9.19.3.8 setReceiptId()

```
void es.ull.esit.app.middleware.model.Drink.setReceiptId (
            Long receiptId)  [inline]
```

Sets the identifier of the related receipt.

Parameters

receipt↩ Id	[Long] Identifier of the receipt.

Definition at line 121 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

```
00121                                               {
00122     this.receiptId = receiptId;
00123   }
```

References receiptId.

9.19.4 Member Data Documentation

9.19.4.1 drinksId

```
Long es.ull.esit.app.middleware.model.Drink.drinksId  [private]
```

Unique identifier of the drink (JSON property "drinksId").

Definition at line 15 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

Referenced by Drink(), getDrinksId(), and setDrinksId().

9.19.4.2 drinksPrice

```
Integer es.ull.esit.app.middleware.model.Drink.drinksPrice  [private]
```

Price of the drink (JSON property "drinksPrice").

Definition at line 23 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

Referenced by Drink(), getDrinksPrice(), and setDrinksPrice().

9.19.4.3 itemDrinks

```
String es.ull.esit.app.middleware.model.Drink.itemDrinks  [private]
```

Name of the drink item (JSON property "itemDrinks").

Definition at line 19 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

Referenced by Drink(), getItemDrinks(), and setItemDrinks().

9.19.4.4 receiptId

```
Long es.ull.esit.app.middleware.model.Drink.receiptId  [private]
```

Identifier of the receipt this drink belongs to (JSON property "receiptId").

Currently not provided by the backend.

Definition at line 30 of file src/main/java/es/ull/esit/app/middleware/model/Drink.java.

Referenced by Drink(), getReceiptId(), and setReceiptId().

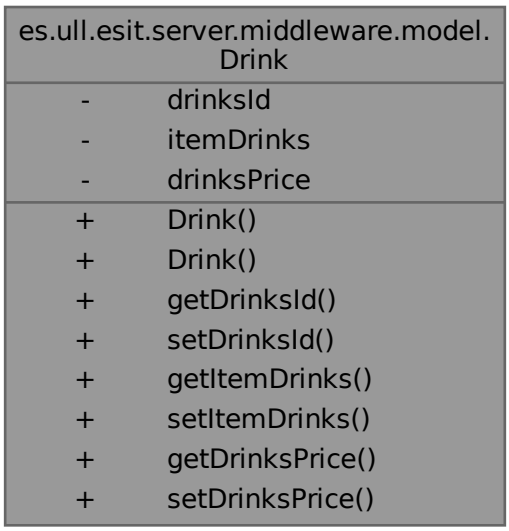
The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/app/middleware/model/Drink.java

9.20 es.ull.esit.server.middleware.model.Drink Class Reference

JPA entity that represents a drink in the menu.

Collaboration diagram for es.ull.esit.server.middleware.model.Drink:



Public Member Functions

- Drink ()

  Default constructor required by JPA.
- Drink (Long drinksId, String itemDrinks, Integer drinksPrice)

  Full constructor.
- Long getDrinksId ()

  Gets the drink identifier.
- void setDrinksId (Long drinksId)

  Sets the drink identifier.
- String getItemDrinks ()

  Gets the drink name.
- void setItemDrinks (String itemDrinks)

  Sets the drink name.
- Integer getDrinksPrice ()

  Gets the drink price.
- void setDrinksPrice (Integer drinksPrice)

  Sets the drink price.

Private Attributes

- Long drinksId

  Primary key of the drink (column "id").
- String itemDrinks

  Name of the drink (column "name").
- Integer drinksPrice

  Price of the drink in integer units (column "price").

9.20.1 Detailed Description

JPA entity that represents a drink in the menu.

```
It is mapped to the "drinks" table used by the REST API.
Each row contains an auto generated identifier, a name and a price.
```

Definition at line 15 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

9.20.2 Constructor & Destructor Documentation

9.20.2.1 Drink() [1/2]

```
es.ull.esit.server.middleware.model.Drink.Drink ()  [inline]
```

Default constructor required by JPA.

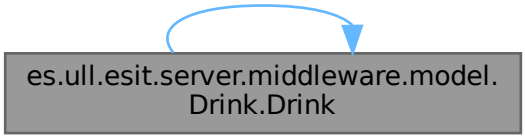
Definition at line 34 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00034                 {
00035   }
```

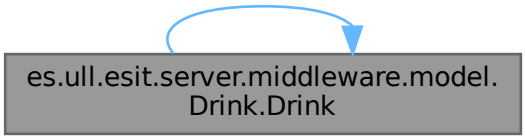
References Drink().

Referenced by Drink().

Here is the call graph for this function:



Here is the caller graph for this function:



9.20.2.2 Drink() [2/2]

```
es.ull.esit.server.middleware.model.Drink.Drink (
            Long drinksId,
            String itemDrinks,
            Integer drinksPrice)  [inline]
```

Full constructor.

Parameters

drinksId	[Long] Identifier of the drink.
itemDrinks	[String] Name of the drink.
drinksPrice	[Integer] Price of the drink.

Definition at line 44 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00044                                                                    {
00045     this.drinksId = drinksId;
00046     this.itemDrinks = itemDrinks;
00047     this.drinksPrice = drinksPrice;
00048   }
```

References drinksId, drinksPrice, and itemDrinks.

9.20.3 Member Function Documentation

9.20.3.1 getDrinksId()

```
Long es.ull.esit.server.middleware.model.Drink.getDrinksId ()  [inline]
```

Gets the drink identifier.

Returns

[Long] Drink id.

Definition at line 55 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00055                           {
00056     return drinksId;
00057   }
```

References drinksId.

9.20.3.2 getDrinksPrice()

```
Integer es.ull.esit.server.middleware.model.Drink.getDrinksPrice ()  [inline]
```

Gets the drink price.

Returns

[Integer] Drink price.

Definition at line 93 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00093                                {
00094     return drinksPrice;
00095   }
```

References drinksPrice.

9.20.3.3 getItemDrinks()

```
String es.ull.esit.server.middleware.model.Drink.getItemDrinks ()  [inline]
```

Gets the drink name.

Returns

[String] Drink name.

Definition at line 75 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00075                               {
00076     return itemDrinks;
00077   }
```

References itemDrinks.

9.20.3.4 setDrinksId()

```
void es.ull.esit.server.middleware.model.Drink.setDrinksId (
            Long drinksId)  [inline]
```

Sets the drink identifier.

Generated by the database.

Parameters

drinksId	[Long] Drink id.

Definition at line 66 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00066                                    {
00067     this.drinksId = drinksId;
00068   }
```


References drinksId.

Parameters

drinksPrice	[Integer] New price of the drink.

Definition at line 102 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00102                                                {
00103     this.drinksPrice = drinksPrice;
00104   }
```

References drinksPrice.

9.20.3.6 setItemDrinks()

```
void es.ull.esit.server.middleware.model.Drink.setItemDrinks (
             String itemDrinks)  [inline]
```

Sets the drink name.

Parameters

itemDrinks	[String] New name of the drink.

Definition at line 84 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

```
00084                                               {
00085     this.itemDrinks = itemDrinks;
00086   }
```

References itemDrinks.

9.20.4 Member Data Documentation

9.20.4.1 drinksId

```
Long es.ull.esit.server.middleware.model.Drink.drinksId  [private]
```

Primary key of the drink (column "id").

Definition at line 21 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

Referenced by Drink(), getDrinksId(), and setDrinksId().

9.20.4.2 drinksPrice

```
Integer es.ull.esit.server.middleware.model.Drink.drinksPrice  [private]
```

Price of the drink in integer units (column "price").

Definition at line 29 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

Referenced by Drink(), getDrinksPrice(), and setDrinksPrice().

9.20.4.3 itemDrinks

```
String es.ull.esit.server.middleware.model.Drink.itemDrinks  [private]
```

Name of the drink (column "name").

Definition at line 25 of file server/src/main/java/es/ull/esit/server/middleware/model/Drink.java.

Referenced by Drink(), getItemDrinks(), and setItemDrinks().

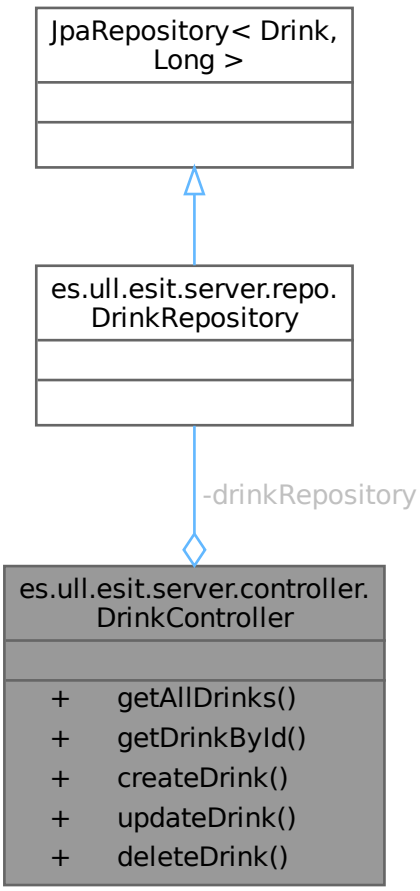
The documentation for this class was generated from the following file:

- server/src/main/java/es/ull/esit/server/middleware/model/Drink.java

9.21 es.ull.esit.server.controller.DrinkController Class Reference

REST controller for managing drinks.

Collaboration diagram for es.ull.esit.server.controller.DrinkController:



Public Member Functions

- ResponseEntity< List< Drink > > getAllDrinks ()
  Returns the complete list of drinks.
- ResponseEntity< Drink > getDrinkById (@PathVariable Long id)
  Returns a single drink by its id.
- ResponseEntity< Drink > createDrink (@RequestBody Drink drink)
  Creates a new drink.
- ResponseEntity< Drink > updateDrink (@PathVariable Long id, @RequestBody Drink drink)
  Updates an existing drink.
- ResponseEntity< Void > deleteDrink (@PathVariable Long id)
  Deletes a drink by its id.

Private Attributes

- DrinkRepository drinkRepository
  Repository used to access the "drinks" table.

9.21.1 Detailed Description

REST controller for managing drinks.

```
Provides CRUD operations for Drink entities using the path
"/api/drinks". These endpoints are used by the Swing client to load
and modify drink information.
```

Definition at line 22 of file DrinkController.java.

9.21.2 Member Function Documentation

9.21.2.1 createDrink()

```
ResponseEntity< Drink > es.ull.esit.server.controller.DrinkController.createDrink (
            @RequestBody Drink drink)  [inline]
```

Creates a new drink.

```
HTTP POST /api/drinks
```

Parameters

drink	[Drink] Drink data sent in the request body.

Returns

[ResponseEntity<Drink>] The created drink with status 201.

Definition at line 65 of file DrinkController.java.

```
00065                                                                    {
00066     Drink saved = drinkRepository.save(drink);
00067     return ResponseEntity.status(HttpStatus.CREATED).body(saved);
00068   }
```

References drinkRepository.

9.21.2.2 deleteDrink()

```
ResponseEntity< Void > es.ull.esit.server.controller.DrinkController.deleteDrink (
            @PathVariable Long id)  [inline]
```

Deletes a drink by its id.

```
HTTP DELETE /api/drinks/{id}
```

Parameters

id	[Long] Identifier of the drink to delete.

Returns

[ResponseEntity<Void>] 204 if deleted or 404 if not found.

Definition at line 99 of file DrinkController.java.

```
00099                                                                    {
00100     if (!drinkRepository.existsById(id)) {
00101       return ResponseEntity.notFound().build();
00102     }
00103     drinkRepository.deleteById(id);
00104     return ResponseEntity.noContent().build();
00105   }
```

References deleteDrink(), and drinkRepository.

Referenced by deleteDrink().

Here is the call graph for this function:

Here is the caller graph for this function:

9.21.2.3 getAllDrinks()

```
ResponseEntity< List< Drink > > es.ull.esit.server.controller.DrinkController.getAllDrinks ()
[inline]
```

Returns the complete list of drinks.

```
HTTP GET /api/drinks
```

Returns

[ResponseEntity<List<Drink>>] List of drinks with status 200.

Definition at line 36 of file DrinkController.java.

```
00036                                                {
00037     List<Drink> drinks = drinkRepository.findAll();
00038     return ResponseEntity.ok(drinks);
00039   }
```

References drinkRepository.

9.21.2.4 getDrinkById()

```
ResponseEntity< Drink > es.ull.esit.server.controller.DrinkController.getDrinkById (
            @PathVariable Long id)  [inline]
```

Returns a single drink by its id.

```
HTTP GET /api/drinks/{id}
```

Parameters

id	[Long] Identifier of the drink.

Returns

[ResponseEntity<Drink>] The drink if found or 404 otherwise.

Definition at line 50 of file DrinkController.java.

```
00050                                                                {
00051     Optional<Drink> drink = drinkRepository.findById(id);
00052     return drink.map(ResponseEntity::ok)
00053         .orElseGet(() -> ResponseEntity.notFound().build());
00054   }
```

References drinkRepository, and getDrinkById().

Referenced by getDrinkById().

Here is the call graph for this function:

Here is the caller graph for this function:

9.21.2.5 updateDrink()

```
ResponseEntity< Drink > es.ull.esit.server.controller.DrinkController.updateDrink (
            @PathVariable Long id,
            @RequestBody Drink drink)  [inline]
```

Updates an existing drink.

```
HTTP PUT /api/drinks/{id}
```

Parameters

id	[Long] Identifier of the drink to update.
drink	[Drink] New data for the drink.

Returns

[ResponseEntity<Drink>] The updated drink or 404 if not found.

Definition at line 80 of file DrinkController.java.

```
00081                                {
00082     if (!drinkRepository.existsById(id)) {
00083       return ResponseEntity.notFound().build();
00084     }
00085     drink.setDrinksId(id);
00086     Drink updated = drinkRepository.save(drink);
00087     return ResponseEntity.ok(updated);
00088   }
```

References drinkRepository, and updateDrink().

Referenced by updateDrink().

Here is the call graph for this function:

Here is the caller graph for this function:

9.21.3 Member Data Documentation

9.21.3.1 drinkRepository

```
DrinkRepository es.ull.esit.server.controller.DrinkController.drinkRepository  [private]
```

Repository used to access the "drinks" table.

Definition at line 26 of file DrinkController.java.

Referenced by createDrink(), deleteDrink(), getAllDrinks(), getDrinkById(), and updateDrink().

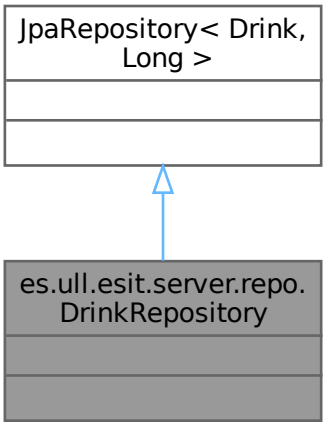
The documentation for this class was generated from the following file:

- DrinkController.java

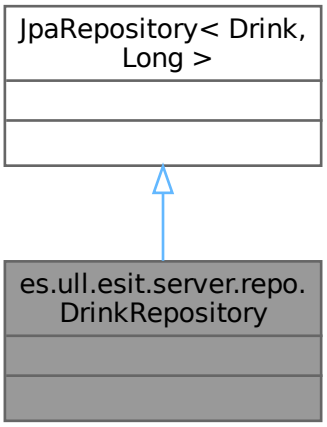
9.22 es.ull.esit.server.repo.DrinkRepository Interface Reference

Repository interface for managing drinks in the database.

Inheritance diagram for es.ull.esit.server.repo.DrinkRepository:



Collaboration diagram for es.ull.esit.server.repo.DrinkRepository:



9.22.1 Detailed Description

Repository interface for managing drinks in the database.

```
Extends JpaRepository to provide basic CRUD operations on the "drinks"
table, such as findAll, findById, save and delete.
```

Definition at line 12 of file DrinkRepository.java.

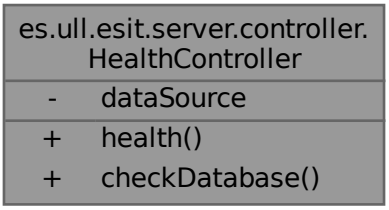
The documentation for this interface was generated from the following file:

- DrinkRepository.java

9.23 es.ull.esit.server.controller.HealthController Class Reference

REST controller for health and database connectivity checks.

Collaboration diagram for es.ull.esit.server.controller.HealthController:



Public Member Functions

- ResponseEntity< Map< String, Object > > health ()

  Basic health endpoint.
- ResponseEntity< Map< String, Object > > checkDatabase ()

  Database connectivity check.

Private Attributes

- DataSource dataSource

  DataSource used to verify connectivity with the database.

9.23.1 Detailed Description

REST controller for health and database connectivity checks.

```
Exposes simple diagnostic endpoints to:
- receive basic service liveness information.
- check connectivity with the underlying database.
Not required by the Swing frontend but are very
useful for debugging, monitoring or for external tools that need to
verify that the service and the database are up and running.
```

Definition at line 27 of file HealthController.java.

9.23.2 Member Function Documentation

9.23.2.1 checkDatabase()

```
ResponseEntity< Map< String, Object > > es.ull.esit.server.controller.HealthController.↩
checkDatabase ()  [inline]
```

Database connectivity check.

```
Endpoint: GET /api/db-check

Tries to obtain a JDBC connection from the configured DataSource
and verifies that it is valid. If the connection is valid, information
about the database catalog and URL is included in the response.

On success:
- HTTP 200 with JSON fields "status" = "UP" and "database" =
"Connected".

On failure:
- HTTP 503 (SERVICE_UNAVAILABLE) with "status" = "DOWN" and an
additional "error" message describing the issue.
```

Returns

[ResponseEntity<Map<String, Object>>] Database connectivity status: HTTP 200 if connected, HTTP 503 otherwise.

Definition at line 73 of file HealthController.java.

```
00073                                                       {
00074     Map<String, Object> response = new HashMap<>();
00075
00076     try (Connection conn = dataSource.getConnection()) {
00077       boolean isValid = conn.isValid(2);
00078
00079       if (isValid) {
00080         response.put("status", "UP");
00081         response.put("database", "Connected");
00082         response.put("catalog", conn.getCatalog());
00083         response.put("url", conn.getMetaData().getURL());
00084         response.put("timestamp", System.currentTimeMillis());
00085         return ResponseEntity.ok(response);
00086       } else {
00087         response.put("status", "DOWN");
00088         response.put("database", "Connection not valid");
00089         return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE).body(response);
00090       }
00091     } catch (Exception e) {
00092       response.put("status", "DOWN");
00093       response.put("database", "Connection failed");
00094       response.put("error", e.getMessage());
00095       response.put("timestamp", System.currentTimeMillis());
00096       return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE).body(response);
00097     }
00098   }
```

References checkDatabase(), and dataSource.

Referenced by checkDatabase().

Here is the call graph for this function:

Here is the caller graph for this function:

9.23.2.2 health()

```
ResponseEntity< Map< String, Object > > es.ull.esit.server.controller.HealthController.health
() [inline]
```

Basic health endpoint.

```
Endpoint: GET /api/health

Returns a small JSON payload indicating that the service is up,
together with a timestamp and a human-readable service name.
```

Returns

[ResponseEntity<Map<String, Object>>] Health status with HTTP 200.

Definition at line 44 of file HealthController.java.

```
00044                                                {
00045     Map<String, Object> response = new HashMap<>();
00046     response.put("status", "UP");
00047     response.put("timestamp", System.currentTimeMillis());
00048     response.put("service", "Restaurant Server");
00049     return ResponseEntity.ok(response);
00050   }
```

References health().

Referenced by health().

Here is the call graph for this function:

Here is the caller graph for this function:

9.23.3 Member Data Documentation

9.23.3.1 dataSource

```
DataSource es.ull.esit.server.controller.HealthController.dataSource  [private]
```

DataSource used to verify connectivity with the database.

Definition at line 31 of file HealthController.java.

Referenced by checkDatabase().

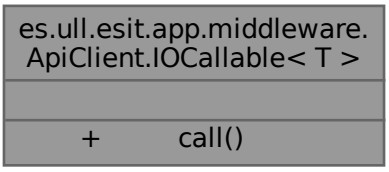
The documentation for this class was generated from the following file:

- HealthController.java

9.24 es.ull.esit.app.middleware.ApiClient.IOCallable< T > Interface Template Reference

Functional interface for lambdas that may throw InterruptedException or IOException.

Collaboration diagram for es.ull.esit.app.middleware.ApiClient.IOCallable< T >:



Public Member Functions

- T call () throws InterruptedException, IOException

9.24.1 Detailed Description

Functional interface for lambdas that may throw InterruptedException or IOException.

Definition at line 96 of file ApiClient.java.

9.24.2 Member Function Documentation

9.24.2.1 call()

```
T es.ull.esit.app.middleware.ApiClient.IOCallable< T >.call () throws InterruptedException,
IOException
```

The documentation for this interface was generated from the following file:

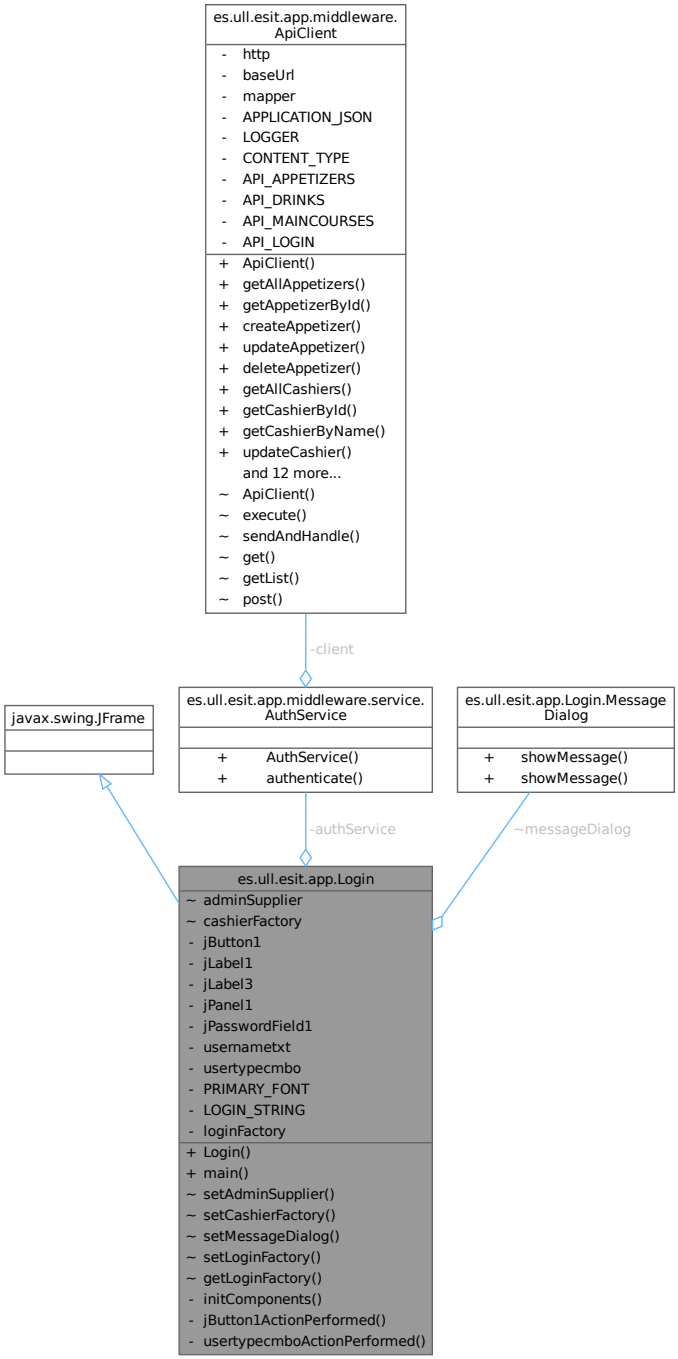
- ApiClient.java

9.25 es.ull.esit.app.Login Class Reference

Login window for the Restaurant System.

Inheritance diagram for es.ull.esit.app.Login:

Collaboration diagram for es.ull.esit.app.Login:



Classes

- interface MessageDialog

Public Member Functions

- Login ()

  Constructor.

Static Public Member Functions

- static void main (String[ ] args)

  Main entry point to start the login window application.

Private Member Functions

- void initComponents ()

  Initializes GUI components.
- void jButton1ActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the "Log In" button.
- void usertypecmboActionPerformed (java.awt.event.ActionEvent evt)

  Action handler for the user type combo box.

Private Attributes

- final transient AuthService authService

  Service to handle authentication logic.
- javax.swing.JButton jButton1

  Button to perform login ("Log In").
- javax.swing.JLabel jLabel1

  Label for the image/logo.
- javax.swing.JLabel jLabel3

  Label for the title ("User Login").
- javax.swing.JPanel jPanel1

  Main panel container.
- javax.swing.JPasswordField jPasswordField1

  Input field for the password.
- javax.swing.JTextField usernametxt

  Input field for the username.
- javax.swing.JComboBox< String > usertypecmbo

  Combo box for user type (admin/cashier).

Static Private Attributes

- static final String PRIMARY_FONT = "Yu Gothic UI"

  Primary font used in the GUI.
- static final String LOGIN_STRING = "Log in"

  String constant for "Log in" button text.
- static java.util.function.Supplier< Login > loginFactory = Login::new

9.25.1 Detailed Description

Login window for the Restaurant System.

```
Shows a Swing form that lets the user:
- enter his/her/their username.
- enter his/her/their password.
- select user type (admin or cashier) from a combo box.
- click the "Log In" button to authenticate.

The credentials are sent to the backend through AuthService and
ApiClient.
According to the role returned by the server (ADMIN or CASHIER), the
application opens the corresponding window: AdminLogin or Cashier.
```

Definition at line 23 of file Login.java.

9.25.2 Constructor & Destructor Documentation

9.25.2.1 Login()

```
es.ull.esit.app.Login.Login ()  [inline]
```

Constructor.

Creates the login window, initializes the AuthService and GUI components.

The ApiClient is configured to connect to the backend at "http://localhost:8080", which must match the URL of the Spring Boot backend server.

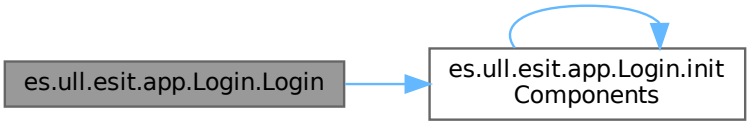
Definition at line 78 of file Login.java.

```
00078                 {
00079     initComponents();
00080     ApiClient client = new ApiClient("http://localhost:8080");
00081     this.authService = new AuthService(client);
00082   }
```

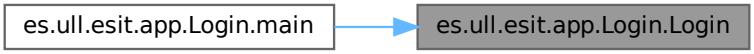
References initComponents().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



9.25.3 Member Function Documentation

9.25.3.1 initComponents()

```
void es.ull.esit.app.Login.initComponents ()  [inline], [private]
```

Initializes GUI components.

```
   Generated by the Form Editor: do not modify.
   Sets up logo image, main label, input fields for username and password, combo box for user type, and button
```

Definition at line 92 of file Login.java.

```
00092                               {
00093
00094     jPanel1 = new javax.swing.JPanel();
00095     jLabel1 = new javax.swing.JLabel();
00096     jLabel3 = new javax.swing.JLabel();
00097     usernametxt = new javax.swing.JTextField();
00098     usertypecmbo = new javax.swing.JComboBox<>();
00099     jButton1 = new javax.swing.JButton();
00100     jPasswordField1 = new javax.swing.JPasswordField();
00101
00102     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00103     setTitle("Login");
00104     setResizable(false);
00105
00106     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00107
00108     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
      NOI18N
00109
00110     jLabel3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 24)); // NOI18N
00111     jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00112     jLabel3.setText("User Login ");
00113
00114     usernametxt.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00115     usernametxt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00116     usernametxt.setBorder(javax.swing.BorderFactory.createTitledBorder(
00117         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Username",
00118         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00119         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00120
00121     usertypecmbo.setEditable(true);
00122     usertypecmbo.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00123     usertypecmbo.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "admin", "cashier"
      }));
00124     usertypecmbo.addActionListener(this::usertypecmboActionPerformed);
00125
00126     jButton1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00127     jButton1.setText(LOGIN_STRING);
00128     jButton1.addActionListener(this::jButton1ActionPerformed);
00129
00130     jPasswordField1.setFont(new java.awt.Font("Thonburi", 0, 18)); // NOI18N
00131     jPasswordField1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00132     jPasswordField1.setBorder(javax.swing.BorderFactory.createTitledBorder(
00133         javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)), "Password",
00134         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00135         new java.awt.Font("Lucida Grande", 0, 18))); // NOI18N
00136
00137     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00138     jPanel1.setLayout(jPanel1Layout);
00139     jPanel1Layout.setHorizontalGroup(
00140         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00141             .addGroup(jPanel1Layout.createSequentialGroup()
00142                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00143                     .addGroup(jPanel1Layout.createSequentialGroup()
00144                         .addGap(132, 132, 132)
00145                         .addComponent(jLabel1))
00146                     .addGroup(jPanel1Layout.createSequentialGroup()
00147                         .addGap(61, 61, 61)
00148                         .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 309,
00149                             javax.swing.GroupLayout.PREFERRED_SIZE))
00150                     .addGroup(jPanel1Layout.createSequentialGroup()
00151                         .addGap(74, 74, 74)
00152
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00153                             .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE,
      311,
```

```
00154                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00155                             .addComponent(usernametxt, javax.swing.GroupLayout.PREFERRED_SIZE, 311,
00156                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00157                             .addComponent(usertypecmbo, javax.swing.GroupLayout.PREFERRED_SIZE, 311,
00158                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00159                     .addGroup(jPanel1Layout.createSequentialGroup()
00160                         .addGap(146, 146, 146)
00161                         .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00162                             javax.swing.GroupLayout.PREFERRED_SIZE)))
00163                 .addContainerGap(86, Short.MAX_VALUE)));
00164     jPanel1Layout.setVerticalGroup(
00165         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00166             .addGroup(jPanel1Layout.createSequentialGroup()
00167                 .addGap(39, 39, 39)
00168                 .addComponent(jLabel1)
00169                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00170                 .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
00171                     javax.swing.GroupLayout.PREFERRED_SIZE)
00172                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00173                 .addComponent(usernametxt, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00174                     javax.swing.GroupLayout.PREFERRED_SIZE)
00175                 .addGap(18, 18, 18)
00176                 .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00177                     javax.swing.GroupLayout.PREFERRED_SIZE)
00178                 .addGap(18, 18, 18)
00179                 .addComponent(usertypecmbo, javax.swing.GroupLayout.PREFERRED_SIZE, 46,
00180                     javax.swing.GroupLayout.PREFERRED_SIZE)
00181                 .addGap(18, 18, 18)
00182                 .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00183                     javax.swing.GroupLayout.PREFERRED_SIZE)
00184                 .addContainerGap(39, Short.MAX_VALUE)));
00185 
00186     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00187     getContentPane().setLayout(layout);
00188     layout.setHorizontalGroup(
00189         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00190             .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00191                 javax.swing.GroupLayout.PREFERRED_SIZE));
00192     layout.setVerticalGroup(
00193         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00194             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00195                 Short.MAX_VALUE));
00196 
00197     pack();
00198     setLocationRelativeTo(null);
00199   }// </editor-fold>//GEN-END:initComponents
```

References initComponents(), jButton1, jLabel1, jLabel3, jPanel1, jPasswordField1, LOGIN_STRING, PRIMARY_FONT, usernametxt, and usertypecmbo.

Referenced by initComponents(), and Login().

Here is the call graph for this function:

Here is the caller graph for this function:

9.25.3.2 jButton1ActionPerformed()

```
void es.ull.esit.app.Login.jButton1ActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the "Log In" button.

```
Action steps:
1. Read username and password from the fields.
2. Disable the button and show "Loading..." to avoid double clicks.
3. Run authentication in a background thread.
4. On success, open the admin or cashier window depending on the role.
5. On error, show a dialog and re-enable the button.
```

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by button click.

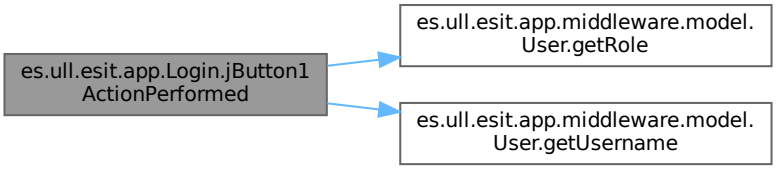
Definition at line 214 of file Login.java.

```
00214                                                    {//
      GEN-FIRST:event_jButton1ActionPerformed
00215     // Get username and password from input fields.
00216     String usernameInput = usernametxt.getText();
00217     String passwordInput = new String(jPasswordField1.getPassword());
00218
00219     jButton1.setEnabled(false);
00220     jButton1.setText("Loading...");
00221
00222     new Thread(() -> {
00223       try {
00224         // Authenticate user through AuthService.
00225         User loggedInUser = authService.authenticate(usernameInput, passwordInput);
00226
00227         SwingUtilities.invokeLater(() -> {
00228
00229           // Open the corresponding window based on user role. Use the
00230           // injectable factories so tests can substitute frames.
00231           if ("ADMIN".equalsIgnoreCase(loggedInUser.getRole())) {
00232             adminSupplier.get().setVisible(true);
00233           } else if ("CASHIER".equalsIgnoreCase(loggedInUser.getRole())) {
00234             cashierFactory.apply(loggedInUser.getUsername()).setVisible(true);
00235           } else {
00236             messageDialog.showMessage(this, "Unknown Role: " + loggedInUser.getRole());
00237             jButton1.setEnabled(true);
00238             jButton1.setText(LOGIN_STRING);
00239             return;
00240           }
00241
00242           this.dispose();
00243         });
00244
00245       } catch (Exception e) {
00246         SwingUtilities.invokeLater(() -> {
00247           messageDialog.showMessage(this, "Login failed: " + e.getMessage(), "Login Error",
      JOptionPane.ERROR_MESSAGE);
```

```
00248
00249           jButton1.setEnabled(true);
00250           jButton1.setText(LOGIN_STRING);
00251         });
00252       }
00253     }).start();
00254   }// GEN-LAST:event_jButton1ActionPerformed
```

References authService, es.ull.esit.app.middleware.model.User.getRole(), es.ull.esit.app.middleware.model.User.getUsername(), jButton1, jPasswordField1, LOGIN_STRING, and usernametxt.

Here is the call graph for this function:



9.25.3.3 main()

```
void es.ull.esit.app.Login.main (
             String[] args)  [inline], [static]
```

Main entry point to start the login window application.

Parameters

args	[String[]] Command line arguments (not used).

Definition at line 273 of file Login.java.

```
00273                                       {
00274     try {
00275       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00276         if ("Nimbus".equals(info.getName())) {
00277           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00278           break;
00279         }
00280       }
00281     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00282         | javax.swing.UnsupportedLookAndFeelException ex) {
00283       java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00284     }
00285
00286     java.awt.EventQueue.invokeLater(() -> getLoginFactory().get().setVisible(true));
00287   }
```

References Login().

Here is the call graph for this function:

9.25.3.4 usertypecmboActionPerformed()

```
void es.ull.esit.app.Login.usertypecmboActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Action handler for the user type combo box.

```
The selection is not used in this implementation, as the role is
determined by the server.
```

Parameters

evt	[java.awt.event.ActionEvent] Action event triggered by combo box selection.

Definition at line 264 of file Login.java.

```
00264                                                                    {//
      GEN-FIRST:event_usertypecmboActionPerformed
00265     // No action needed here.
00266   }// GEN-LAST:event_usertypecmboActionPerformed
```

9.25.4 Member Data Documentation

9.25.4.1 authService

```
final transient AuthService es.ull.esit.app.Login.authService  [private]
```

Service to handle authentication logic.

Definition at line 26 of file Login.java.

Referenced by jButton1ActionPerformed().

9.25.4.2 jButton1

```
javax.swing.JButton es.ull.esit.app.Login.jButton1  [private]
```

Button to perform login ("Log In").

Definition at line 304 of file Login.java.

Referenced by initComponents(), and jButton1ActionPerformed().

9.25.4.3 jLabel1

```
javax.swing.JLabel es.ull.esit.app.Login.jLabel1  [private]
```

Label for the image/logo.

Definition at line 306 of file Login.java.

Referenced by initComponents().

9.25.4.4 jLabel3

```
javax.swing.JLabel es.ull.esit.app.Login.jLabel3  [private]
```

Label for the title ("User Login").

Definition at line 308 of file Login.java.

Referenced by initComponents().

9.25.4.5 jPanel1

```
javax.swing.JPanel es.ull.esit.app.Login.jPanel1  [private]
```

Main panel container.

Definition at line 310 of file Login.java.

Referenced by initComponents().

9.25.4.6 jPasswordField1

```
javax.swing.JPasswordField es.ull.esit.app.Login.jPasswordField1  [private]
```

Input field for the password.

Definition at line 312 of file Login.java.

Referenced by initComponents(), and jButton1ActionPerformed().

9.25.4.7 LOGIN_STRING

```
final String es.ull.esit.app.Login.LOGIN_STRING = "Log in"  [static], [private]
```

String constant for "Log in" button text.

Definition at line 67 of file Login.java.

Referenced by initComponents(), and jButton1ActionPerformed().

9.25.4.8 loginFactory

```
java.util.function.Supplier<Login> es.ull.esit.app.Login.loginFactory = Login::new  [static],
[private]
```

Definition at line 290 of file Login.java.

9.25.4.9 PRIMARY_FONT

```
final String es.ull.esit.app.Login.PRIMARY_FONT = "Yu Gothic UI"  [static], [private]
```

Primary font used in the GUI.

Definition at line 64 of file Login.java.

Referenced by initComponents().

9.25.4.10 usernametxt

```
javax.swing.JTextField es.ull.esit.app.Login.usernametxt  [private]
```

Input field for the username.

Definition at line 314 of file Login.java.

Referenced by initComponents(), and jButton1ActionPerformed().

9.25.4.11 usertypecmbo

```
javax.swing.JComboBox<String> es.ull.esit.app.Login.usertypecmbo  [private]
```

Combo box for user type (admin/cashier).

Currently not trusted for security decisions.

Definition at line 319 of file Login.java.

Referenced by initComponents().

The documentation for this class was generated from the following file:

- Login.java

9.26 es.ull.esit.server.controller.AuthController.LoginRequest Class Reference

Simple DTO (Data Transfer Object) for login requests payload.

Collaboration diagram for es.ull.esit.server.controller.AuthController.LoginRequest:

Public Attributes

- String username
- String password

9.26.1 Detailed Description

Simple DTO (Data Transfer Object) for login requests payload.

```
It is populated automatically from the JSON request body of the HTTP
request.
```

Definition at line 42 of file AuthController.java.

9.26.2 Member Data Documentation

9.26.2.1 password

```
String es.ull.esit.server.controller.AuthController.LoginRequest.password
```

Definition at line 44 of file AuthController.java.

9.26.2.2 username

```
String es.ull.esit.server.controller.AuthController.LoginRequest.username
```

Definition at line 43 of file AuthController.java.

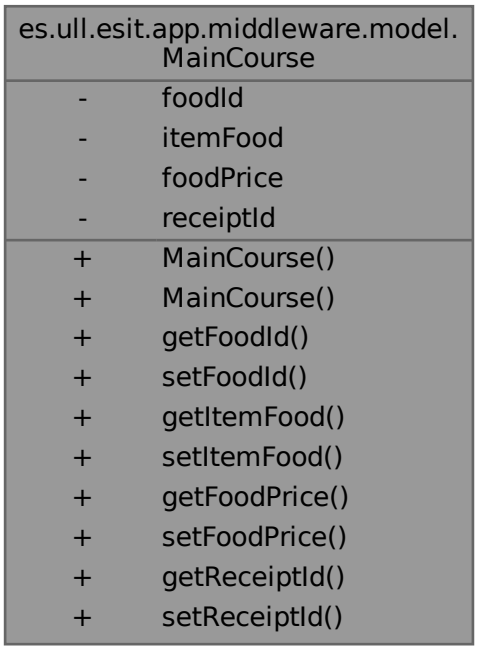
The documentation for this class was generated from the following file:

- AuthController.java

9.27 es.ull.esit.app.middleware.model.MainCourse Class Reference

Client-side model representing a main course returned by the backend.

Collaboration diagram for es.ull.esit.app.middleware.model.MainCourse:



Public Member Functions

- MainCourse ()
  Default constructor required for JSON deserialization.
- MainCourse (Long foodId, String itemFood, Integer foodPrice, Long receiptId)
  Constructs a main course with all fields.
- Long getFoodId ()
  Gets the dish identifier.
- void setFoodId (Long foodId)
  Sets the dish identifier.
- String getItemFood ()
  Gets the name of the dish.
- void setItemFood (String itemFood)
  Sets the name of the dish.
- Integer getFoodPrice ()
  Gets the price of the dish.
- void setFoodPrice (Integer foodPrice)
  Sets the price of the dish.
- Long getReceiptId ()
  Gets the identifier of the related receipt.
- void setReceiptId (Long receiptId)
  Sets the identifier of the related receipt.

Private Attributes

- Long foodId

  Unique identifier of the dish (JSON property "foodId").
- String itemFood

  Name of the dish (JSON property "itemFood").
- Integer foodPrice

  Price of the dish (JSON property "foodPrice").
- Long receiptId

  Identifier of the receipt this dish belongs to (JSON property "receiptId").

9.27.1 Detailed Description

Client-side model representing a main course returned by the backend.

```
Describes a single dish in the main course category as exposed by the
"/api/maincourses" endpoint.
```

Definition at line 11 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

9.27.2 Constructor & Destructor Documentation

9.27.2.1 MainCourse() [1/2]

```
es.ull.esit.app.middleware.model.MainCourse.MainCourse ()  [inline]
```

Default constructor required for JSON deserialization.

Definition at line 35 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00035                       {
00036   }
```

References MainCourse().

Referenced by MainCourse().

Here is the call graph for this function:

Here is the caller graph for this function:

9.27.2.2 MainCourse() [2/2]

```
es.ull.esit.app.middleware.model.MainCourse.MainCourse (
            Long foodId,
            String itemFood,
            Integer foodPrice,
            Long receiptId)  [inline]
```

Constructs a main course with all fields.

Parameters

foodId	[Long] Unique identifier of the dish.
itemFood	[String] Name of the dish.
foodPrice	[Integer] Price of the dish.
receiptId	[Long] Identifier of the related receipt (optional).

Definition at line 46 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00046                                                                                {
00047         this.foodId = foodId;
00048         this.itemFood = itemFood;
00049         this.foodPrice = foodPrice;
00050         this.receiptId = receiptId;
00051     }
```

References foodId, foodPrice, itemFood, and receiptId.

9.27.3 Member Function Documentation

9.27.3.1 getFoodId()

```
Long es.ull.esit.app.middleware.model.MainCourse.getFoodId ()  [inline]
```

Gets the dish identifier.

Returns

[Long] Unique identifier of the dish.

Definition at line 58 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00058                           {
00059     return foodId;
00060   }
```

References foodId.

9.27.3.2 getFoodPrice()

```
Integer es.ull.esit.app.middleware.model.MainCourse.getFoodPrice ()  [inline]
```

Gets the price of the dish.

Returns

[Integer] Price of the dish.

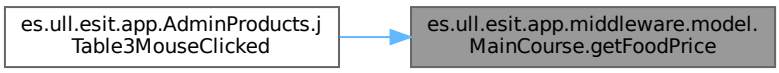
Definition at line 94 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00094                               {
00095     return foodPrice;
00096   }
```

References foodPrice.

Referenced by es.ull.esit.app.AdminProducts.jTable3MouseClicked().

Here is the caller graph for this function:



9.27.3.3 getItemFood()

```
String es.ull.esit.app.middleware.model.MainCourse.getItemFood ()  [inline]
```

Gets the name of the dish.

Returns

[String] Name of the dish.

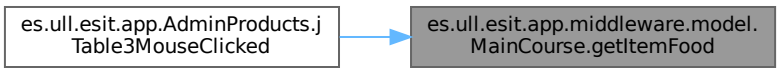
Definition at line 76 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00076                                {
00077     return itemFood;
00078   }
```

References itemFood.

Referenced by es.ull.esit.app.AdminProducts.jTable3MouseClicked().

Here is the caller graph for this function:



9.27.3.4 getReceiptId()

```
Long es.ull.esit.app.middleware.model.MainCourse.getReceiptId ()  [inline]
```

Gets the identifier of the related receipt.

Returns

[Long] Identifier of the receipt or null if not set.

Definition at line 112 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00112                                {
00113     return receiptId;
00114   }
```

References receiptId.

9.27.3.5 setFoodId()

```
void es.ull.esit.app.middleware.model.MainCourse.setFoodId (
             Long foodId)  [inline]
```

Sets the dish identifier.

Parameters

foodId	[Long] Unique identifier of the dish.

Definition at line 67 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00067                                    {
00068     this.foodId = foodId;
00069   }
```

References foodId.

9.27.3.6 setFoodPrice()

```
void es.ull.esit.app.middleware.model.MainCourse.setFoodPrice (
            Integer foodPrice)  [inline]
```

Sets the price of the dish.

Parameters

foodPrice	[Integer] Price of the dish.

Definition at line 103 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00103                                                {
00104     this.foodPrice = foodPrice;
00105   }
```

References foodPrice.

9.27.3.7 setItemFood()

```
void es.ull.esit.app.middleware.model.MainCourse.setItemFood (
            String itemFood)  [inline]
```

Sets the name of the dish.

Parameters

itemFood	[String] Name of the dish.

Definition at line 85 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00085                                               {
00086     this.itemFood = itemFood;
00087   }
```

References itemFood.

9.27.3.8 setReceiptId()

```
void es.ull.esit.app.middleware.model.MainCourse.setReceiptId (
            Long receiptId)  [inline]
```

Sets the identifier of the related receipt.

Parameters

receipt↩ Id	[Long] Identifier of the receipt.

Definition at line 121 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

```
00121                                               {
00122     this.receiptId = receiptId;
00123   }
```

References receiptId.

9.27.4 Member Data Documentation

9.27.4.1 foodId

```
Long es.ull.esit.app.middleware.model.MainCourse.foodId  [private]
```

Unique identifier of the dish (JSON property "foodId").

Definition at line 15 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

Referenced by getFoodId(), MainCourse(), and setFoodId().

9.27.4.2 foodPrice

```
Integer es.ull.esit.app.middleware.model.MainCourse.foodPrice  [private]
```

Price of the dish (JSON property "foodPrice").

Definition at line 23 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

Referenced by getFoodPrice(), MainCourse(), and setFoodPrice().

9.27.4.3 itemFood

```
String es.ull.esit.app.middleware.model.MainCourse.itemFood  [private]
```

Name of the dish (JSON property "itemFood").

Definition at line 19 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

Referenced by getItemFood(), MainCourse(), and setItemFood().

9.27.4.4 receiptId

```
Long es.ull.esit.app.middleware.model.MainCourse.receiptId  [private]
```

Identifier of the receipt this dish belongs to (JSON property "receiptId").

Currently not provided by the backend.

Definition at line 30 of file src/main/java/es/ull/esit/app/middleware/model/MainCourse.java.

Referenced by getReceiptId(), MainCourse(), and setReceiptId().

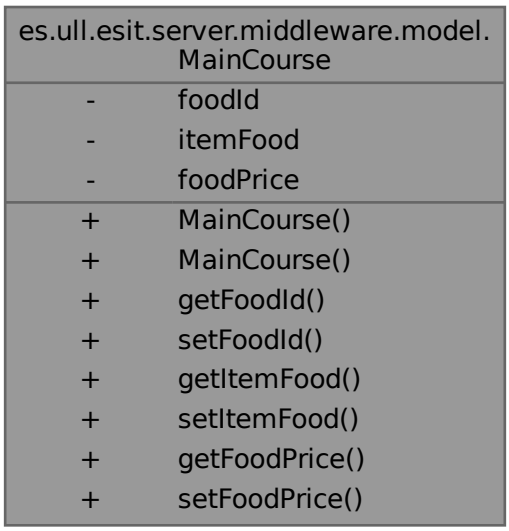
The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/app/middleware/model/MainCourse.java

9.28 es.ull.esit.server.middleware.model.MainCourse Class Reference

JPA entity that represents a main course in the menu.

Collaboration diagram for es.ull.esit.server.middleware.model.MainCourse:



Public Member Functions

- MainCourse ()

  Default constructor required by JPA.
- MainCourse (Long foodId, String itemFood, Integer foodPrice)

  Full constructor.
- Long getFoodId ()

  Gets the main course ID.
- void setFoodId (Long foodId)

  Sets the main course ID.
- String getItemFood ()

  Gets the main course name.
- void setItemFood (String itemFood)

  Sets the main course name.
- Integer getFoodPrice ()

  Gets the main course price.
- void setFoodPrice (Integer foodPrice)

  Sets the main course price.

Private Attributes

- Long foodId

  Primary key of the main course (column "id").
- String itemFood

  Name of the main course (column "name").
- Integer foodPrice

  Price of the main course (column "price").

9.28.1 Detailed Description

JPA entity that represents a main course in the menu.

```
It is mapped to the "maincourse" table used by the REST API.
Each record has an auto-increment ID, a name and a price.
```

Definition at line 15 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

9.28.2 Constructor & Destructor Documentation

9.28.2.1 MainCourse() [1/2]

```
es.ull.esit.server.middleware.model.MainCourse.MainCourse ()  [inline]
```

Default constructor required by JPA.

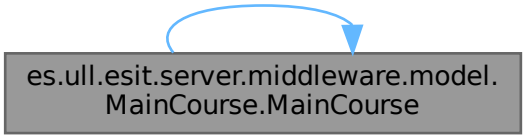
Definition at line 34 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00034                       {
00035   }
```

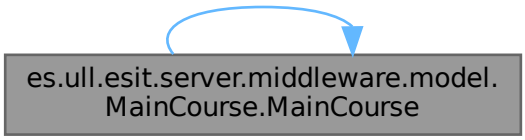
References MainCourse().

Referenced by MainCourse().

Here is the call graph for this function:



Here is the caller graph for this function:



9.28.2.2 MainCourse() [2/2]

```
es.ull.esit.server.middleware.model.MainCourse.MainCourse (
            Long foodId,
            String itemFood,
            Integer foodPrice)  [inline]
```

Full constructor.

Parameters

foodId	[Long] Identifier of the main course.
itemFood	[String] Name of the main course.
foodPrice	[Integer] Price of the main course.

Definition at line 44 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00044                                                                   {
00045     this.foodId = foodId;
00046     this.itemFood = itemFood;
00047     this.foodPrice = foodPrice;
00048   }
```

References foodId, foodPrice, and itemFood.

9.28.3 Member Function Documentation

9.28.3.1 getFoodId()

```
Long es.ull.esit.server.middleware.model.MainCourse.getFoodId ()  [inline]
```

Gets the main course ID.

```
   * @return [Long] Main course id.
```

Definition at line 55 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00055                         {
00056     return foodId;
00057   }
```

References foodId.

9.28.3.2 getFoodPrice()

```
Integer es.ull.esit.server.middleware.model.MainCourse.getFoodPrice ()  [inline]
```

Gets the main course price.

Returns

[Integer] Main course price.

Definition at line 93 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00093                                {
00094     return foodPrice;
00095   }
```

References foodPrice.

9.28.3.3 getItemFood()

```
String es.ull.esit.server.middleware.model.MainCourse.getItemFood ()  [inline]
```

Gets the main course name.

Returns

[String] Main course name.

Definition at line 75 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00075                               {
00076     return itemFood;
00077   }
```

References itemFood.

9.28.3.4 setFoodId()

```
void es.ull.esit.server.middleware.model.MainCourse.setFoodId (
            Long foodId)  [inline]
```

Sets the main course ID.

Generated by the database.

Parameters

foodId	[Long] Main course id.

Definition at line 66 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00066                                    {
00067     this.foodId = foodId;
00068   }
```


References foodId.

Parameters

foodPrice	[Integer] Main course price.

Definition at line 102 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00102                                                {
00103     this.foodPrice = foodPrice;
00104   }
```

References foodPrice.

9.28.3.6 setItemFood()

```
void es.ull.esit.server.middleware.model.MainCourse.setItemFood (
             String itemFood)  [inline]
```

Sets the main course name.

Parameters

itemFood	[String] Main course name.

Definition at line 84 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

```
00084                                               {
00085     this.itemFood = itemFood;
00086   }
```

References itemFood.

9.28.4 Member Data Documentation

9.28.4.1 foodId

```
Long es.ull.esit.server.middleware.model.MainCourse.foodId  [private]
```

Primary key of the main course (column "id").

Definition at line 21 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

Referenced by getFoodId(), MainCourse(), and setFoodId().

9.28.4.2 foodPrice

```
Integer es.ull.esit.server.middleware.model.MainCourse.foodPrice  [private]
```

Price of the main course (column "price").

Definition at line 29 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

Referenced by getFoodPrice(), MainCourse(), and setFoodPrice().

9.28.4.3 itemFood

```
String es.ull.esit.server.middleware.model.MainCourse.itemFood  [private]
```

Name of the main course (column "name").

Definition at line 25 of file server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java.

Referenced by getItemFood(), MainCourse(), and setItemFood().

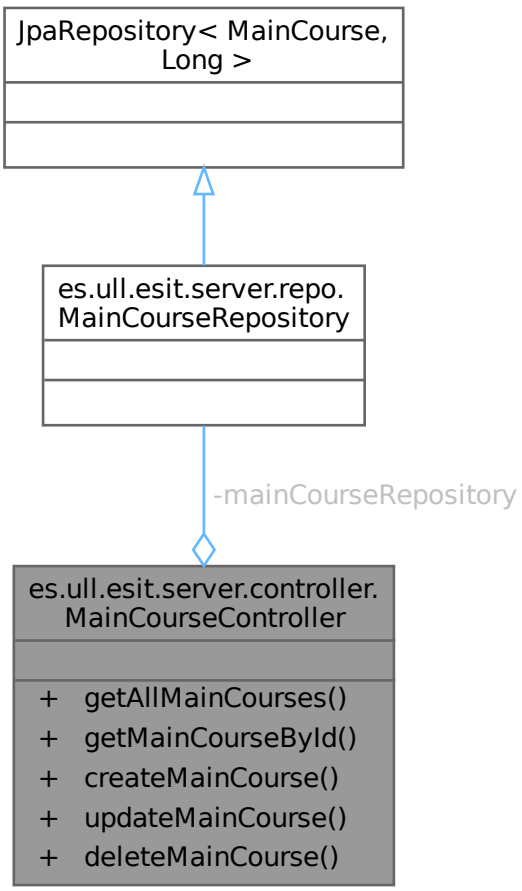
The documentation for this class was generated from the following file:

- server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java

9.29 es.ull.esit.server.controller.MainCourseController Class Reference

REST controller for managing main courses.

Collaboration diagram for es.ull.esit.server.controller.MainCourseController:



Public Member Functions

- ResponseEntity< List< MainCourse > > getAllMainCourses ()

  Returns all available main course items.
- ResponseEntity< MainCourse > getMainCourseById (@PathVariable Long id)

  Returns a specific main course by its ID.
- ResponseEntity< MainCourse > createMainCourse (@RequestBody MainCourse mainCourse)

  Creates a new main course entry.
- ResponseEntity< MainCourse > updateMainCourse (@PathVariable Long id, @RequestBody MainCourse mainCourse)

  Updates an existing main course entry.
- ResponseEntity< Void > deleteMainCourse (@PathVariable Long id)

  Deletes a main course by its ID.

Private Attributes

- MainCourseRepository mainCourseRepository

  Repository used to access the "main_courses" table.

9.29.1 Detailed Description

REST controller for managing main courses.

```
Provides CRUD operations for MainCourse entities using the path
/api/maincourses.
These endpoints are used by the Swing client to load and modify main
course information.
```

Definition at line 22 of file MainCourseController.java.

9.29.2 Member Function Documentation

9.29.2.1 createMainCourse()

```
ResponseEntity< MainCourse > es.ull.esit.server.controller.MainCourseController.createMain↩
Course (
            @RequestBody MainCourse mainCourse)  [inline]
```

Creates a new main course entry.

```
Endpoint: POST /api/maincourses
```

Parameters

mainCourse	[MainCourse] Main course data from the request body.

Returns

[ResponseEntity<MainCourse>] The created main course with status 201

Definition at line 68 of file MainCourseController.java.

```
00068                                                                    {
00069     MainCourse saved = mainCourseRepository.save(mainCourse);
00070     return ResponseEntity.status(HttpStatus.CREATED).body(saved);
00071   }
```

References mainCourseRepository.

9.29.2.2 deleteMainCourse()

```
ResponseEntity< Void > es.ull.esit.server.controller.MainCourseController.deleteMainCourse (
            @PathVariable Long id)  [inline]
```

Deletes a main course by its ID.

```
Endpoint: DELETE /api/maincourses/{id}
```

Parameters

id	[Long] Identifier of the main course to delete.

Returns

[ResponseEntity<Void>] Status 204 if deleted or 404 if not found.

Definition at line 105 of file MainCourseController.java.

```
00105                                                                    {
00106     if (!mainCourseRepository.existsById(id)) {
00107       return ResponseEntity.notFound().build();
00108     }
00109
00110     mainCourseRepository.deleteById(id);
00111     return ResponseEntity.noContent().build();
00112   }
```

References deleteMainCourse(), and mainCourseRepository.

Referenced by deleteMainCourse().

Here is the call graph for this function:

Here is the caller graph for this function:

9.29.2.3 getAllMainCourses()

```
ResponseEntity< List< MainCourse > > es.ull.esit.server.controller.MainCourseController.get↩
AllMainCourses ()  [inline]
```

Returns all available main course items.

```
Endpoint: GET /api/maincourses
```

Returns

[ResponseEntity<List<MainCourse>>] List of main courses with status 200.

Definition at line 37 of file MainCourseController.java.

```
00037                                                          {
00038     List<MainCourse> courses = mainCourseRepository.findAll();
00039     return ResponseEntity.ok(courses);
00040   }
```

References mainCourseRepository.

9.29.2.4 getMainCourseById()

```
ResponseEntity< MainCourse > es.ull.esit.server.controller.MainCourseController.getMain↩
CourseById (
            @PathVariable Long id)  [inline]
```

Returns a specific main course by its ID.

```
Endpoint: GET /api/maincourses/{id}
```

Parameters

id	[Long] Identifier of the main course.

Returns

[ResponseEntity<MainCourse>] The main course if found or 404 otherwise.

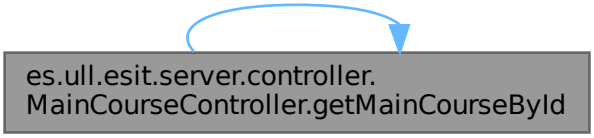
Definition at line 52 of file MainCourseController.java.

```
00052                                                                  {
00053     Optional<MainCourse> course = mainCourseRepository.findById(id);
00054     return course
00055         .map(ResponseEntity::ok)
00056         .orElseGet(() -> ResponseEntity.notFound().build());
00057   }
```

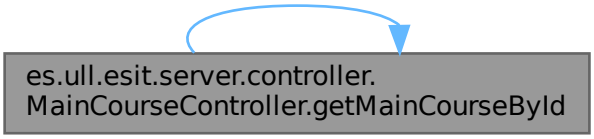
References getMainCourseById(), and mainCourseRepository.

Referenced by getMainCourseById().

Here is the call graph for this function:



Here is the caller graph for this function:



9.29.2.5 updateMainCourse()

```
ResponseEntity< MainCourse > es.ull.esit.server.controller.MainCourseController.updateMain↩
Course (
            @PathVariable Long id,
            @RequestBody MainCourse mainCourse)  [inline]
```

Updates an existing main course entry.

```
Endpoint: PUT /api/maincourses/{id}
```

Parameters

id	[Long] Identifier of the main course to update.
mainCourse	[MainCourse] Updated main course data from the request body.

Returns

[ResponseEntity<MainCourse>] The updated main course if found or 404 otherwise.

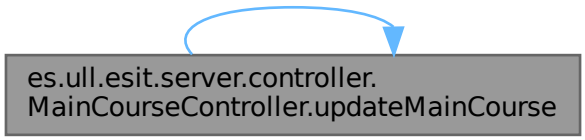
Definition at line 85 of file MainCourseController.java.

```
00085
    {
00086
00087     if (!mainCourseRepository.existsById(id)) {
00088       return ResponseEntity.notFound().build();
00089     }
00090
00091     mainCourse.setFoodId(id);
00092     MainCourse updated = mainCourseRepository.save(mainCourse);
00093     return ResponseEntity.ok(updated);
00094   }
```

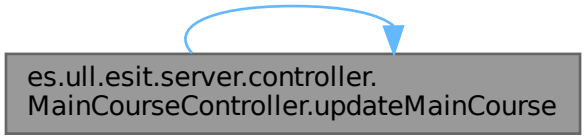
References mainCourseRepository, and updateMainCourse().

Referenced by updateMainCourse().

Here is the call graph for this function:



Here is the caller graph for this function:



9.29.3 Member Data Documentation

9.29.3.1 mainCourseRepository

```
MainCourseRepository es.ull.esit.server.controller.MainCourseController.mainCourseRepository
[private]
```

Repository used to access the "main_courses" table.

Definition at line 26 of file MainCourseController.java.

Referenced by createMainCourse(), deleteMainCourse(), getAllMainCourses(), getMainCourseById(), and updateMainCourse().

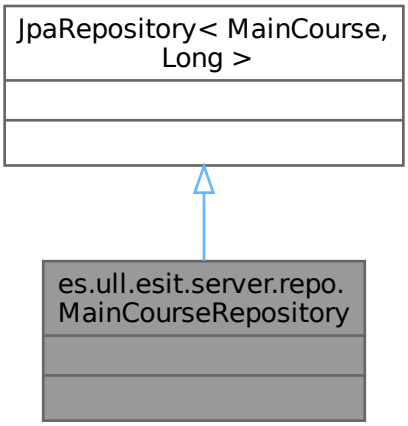
The documentation for this class was generated from the following file:

- MainCourseController.java

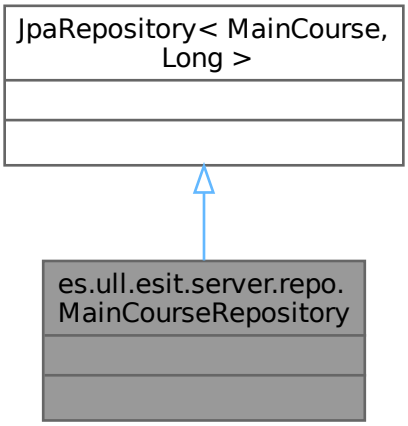
9.30 es.ull.esit.server.repo.MainCourseRepository Interface Reference

Repository interface for managing main courses in the database.

Inheritance diagram for es.ull.esit.server.repo.MainCourseRepository:



Collaboration diagram for es.ull.esit.server.repo.MainCourseRepository:



9.30.1 Detailed Description

Repository interface for managing main courses in the database.

```
Extends JpaRepository to provide basic CRUD operations on the
"main_courses"
table, such as findAll, findById, save and delete.
```

Definition at line 13 of file MainCourseRepository.java.

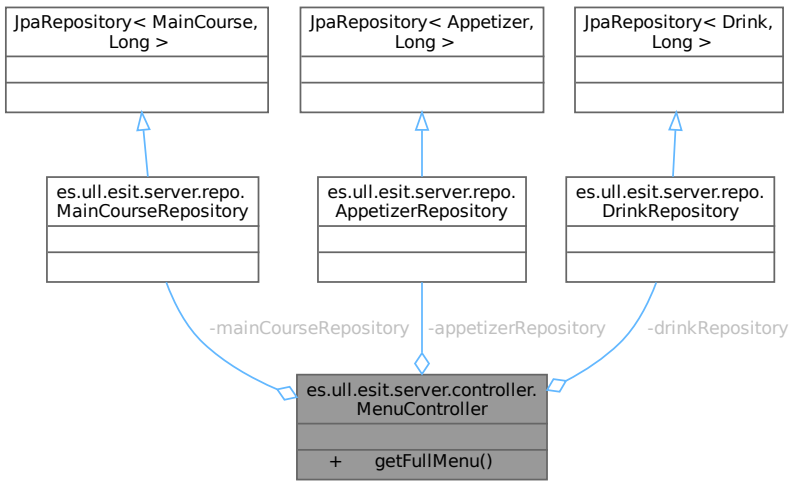
The documentation for this interface was generated from the following file:

- MainCourseRepository.java

9.31 es.ull.esit.server.controller.MenuController Class Reference

REST controller that exposes a consolidated restaurant menu.

Collaboration diagram for es.ull.esit.server.controller.MenuController:



Public Member Functions

- ResponseEntity< Map< String, Object > > getFullMenu ()

  Returns the full menu (main courses, appetizers and drinks).

Private Attributes

- MainCourseRepository mainCourseRepository

  Repository for main course entities.
- AppetizerRepository appetizerRepository

  Repository for appetizer entities.
- DrinkRepository drinkRepository

  Repository for drink entities.

9.31.1 Detailed Description

REST controller that exposes a consolidated restaurant menu.

```
Provides CRUD operation endpoints to retrieve the full menu in a
single
HTTP request. It aggregates main courses, appetizers and drinks into a
single JSON response.
```

Definition at line 29 of file MenuController.java.

9.31.2 Member Function Documentation

9.31.2.1 getFullMenu()

```
ResponseEntity< Map< String, Object > > es.ull.esit.server.controller.MenuController.get↩
FullMenu ()  [inline]
```

Returns the full menu (main courses, appetizers and drinks).

```
Endpoint: GET /api/menu

The response body is a JSON object with the following keys:
- "mainCourses": list of MainCourse
- "appetizers": list of Appetizer
- "drinks": list of Drink
- "totalItems": integer with the total count of menu items
```

Returns

ResponseEntity containing the aggregated menu and HTTP 200.

Definition at line 57 of file MenuController.java.

```
00057                                                       {
00058     Map<String, Object> menu = new HashMap<>();
00059
00060     List<MainCourse> mainCourses = mainCourseRepository.findAll();
00061     List<Appetizer> appetizers = appetizerRepository.findAll();
00062     List<Drink> drinks = drinkRepository.findAll();
00063
00064     menu.put("mainCourses", mainCourses);
00065     menu.put("appetizers", appetizers);
00066     menu.put("drinks", drinks);
00067     menu.put("totalItems", mainCourses.size() + appetizers.size() + drinks.size());
00068
00069     return ResponseEntity.ok(menu);
00070   }
```

References appetizerRepository, drinkRepository, and mainCourseRepository.

9.31.3 Member Data Documentation

9.31.3.1 appetizerRepository

```
AppetizerRepository es.ull.esit.server.controller.MenuController.appetizerRepository  [private]
```

Repository for appetizer entities.

Definition at line 37 of file MenuController.java.

Referenced by getFullMenu().

9.31.3.2 drinkRepository

```
DrinkRepository es.ull.esit.server.controller.MenuController.drinkRepository  [private]
```

Repository for drink entities.

Definition at line 41 of file MenuController.java.

Referenced by getFullMenu().

9.31.3.3 mainCourseRepository

```
MainCourseRepository es.ull.esit.server.controller.MenuController.mainCourseRepository  [private]
```

Repository for main course entities.

Definition at line 33 of file MenuController.java.

Referenced by getFullMenu().

The documentation for this class was generated from the following file:

- MenuController.java

9.32 es.ull.esit.app.Login.MessageDialog Interface Reference

Collaboration diagram for es.ull.esit.app.Login.MessageDialog:

Public Member Functions

- void showMessage (java.awt.Component parent, Object message)
- void showMessage (java.awt.Component parent, Object message, String title, int messageType)

9.32.1 Detailed Description

Definition at line 33 of file Login.java.

9.32.2 Member Function Documentation

9.32.2.1 showMessage() [1/2]

```
void es.ull.esit.app.Login.MessageDialog.showMessage (
            java.awt.Component parent,
            Object message)
```

9.32.2.2 showMessage() [2/2]

```
void es.ull.esit.app.Login.MessageDialog.showMessage (
            java.awt.Component parent,
            Object message,
            String title,
            int messageType)
```

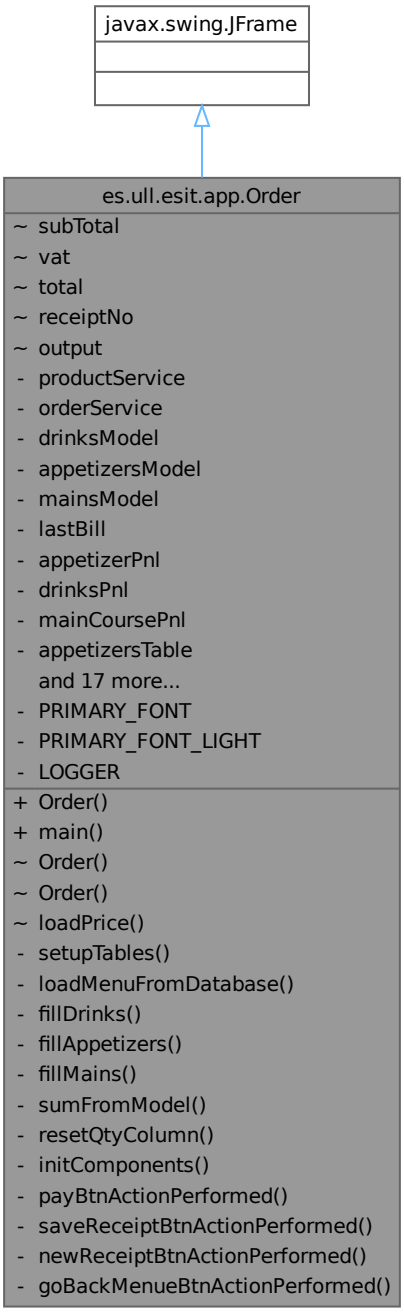
The documentation for this interface was generated from the following file:

- Login.java

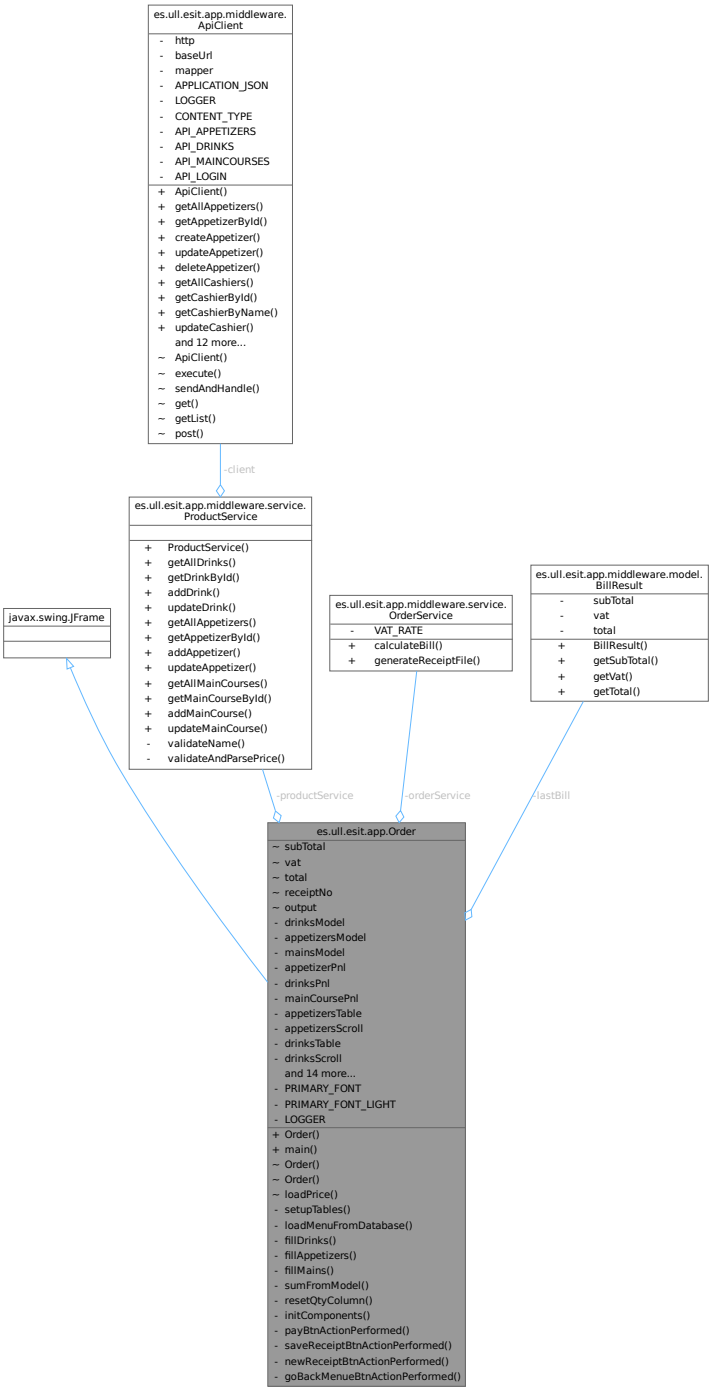
9.33 es.ull.esit.app.Order Class Reference

Main menu window for taking customer orders.

Inheritance diagram for es.ull.esit.app.Order:



Collaboration diagram for es.ull.esit.app.Order:



Public Member Functions

- Order ()

  Constructor.

Static Public Member Functions

- static void main (String[ ] args)

Standalone entry point for testing the Order window.

Private Member Functions

- void setupTables ()

  Configures the three tables (Drinks, Appetizers, Main Course).
- void loadMenuFromDatabase ()

  Loads menu data (drinks, appetizers, main courses) from the backend in a background thread.
- void fillDrinks (java.util.List< Drink > drinks)

  Fills the drinks table model with data from the backend.
- void fillAppetizers (java.util.List< Appetizer > appetizers)

  Fills the appetizers table model with data from the backend.
- void fillMains (java.util.List< MainCourse > mains)

  Fills the main courses table model with data from the backend.
- double sumFromModel (DefaultTableModel model)

  Sums the total price for a given table model.
- void resetQtyColumn (DefaultTableModel model)

  Sets the Qty column of the given model to 0 in all rows.
- void initComponents ()

  Initializes GUI components.
- void payBtnActionPerformed (java.awt.event.ActionEvent evt)

  Calculates subtotal, VAT and total and shows a confirmation dialog.
- void saveReceiptBtnActionPerformed (java.awt.event.ActionEvent evt)

  Saves the current bill information to a text file through OrderService.
- void newReceiptBtnActionPerformed (java.awt.event.ActionEvent evt)

  Starts a new receipt and resets the form to its initial state.
- void goBackMenueBtnActionPerformed (java.awt.event.ActionEvent evt)

  Returns to the Login window.

Private Attributes

- final transient ProductService productService

  Service used to load products from the backend.
- final transient OrderService orderService = new OrderService()

  Service used to calculate bill totals and generate receipt files.
- DefaultTableModel drinksModel

  Table models for the three product categories.
- DefaultTableModel appetizersModel
- DefaultTableModel mainsModel
- transient BillResult lastBill

  Last calculated bill (after pressing Pay).
- javax.swing.JPanel appetizerPnl

  Panel that groups all appetizer items and controls.
- javax.swing.JPanel drinksPnl

  Panel that groups all drink items and controls.
- javax.swing.JPanel mainCoursePnl

  Panel that groups all main course items and controls.
- javax.swing.JTable appetizersTable

  Table for appetizers items.
- javax.swing.JScrollPane appetizersScroll

Scroll pane for appetizers table.

- javax.swing.JTable drinksTable

  Table for drinks items.
- javax.swing.JScrollPane drinksScroll

  Scroll pane for drinks table.
- javax.swing.JButton goBackMenueBtn

  Button to go back to the Login window.
- javax.swing.JLabel jLabel1

  Title label "BLACK PLATE MENU".
- javax.swing.JLabel jLabel2

  Logo / screenshot label on the top left.
- javax.swing.JPanel jPanel1

  Panel that shows subtotal, VAT, total and receipt number.
- javax.swing.JPanel jPanel2

  Main container panel of the Order window.
- javax.swing.JTable mainsTable

  Table for main course items.
- javax.swing.JScrollPane mainsScroll

  Scroll pane for main course table.
- javax.swing.JButton newReceiptBtn

  Button to start a new receipt.
- javax.swing.JButton payBtn

  Button to calculate and confirm payment.
- javax.swing.JLabel receiptNoLbl

  Label that shows the current receipt number.
- javax.swing.JButton saveReceiptBtn

  Button to save the current receipt to a file.
- javax.swing.JLabel subTotalLbl

  Label that displays the subtotal amount.
- javax.swing.JLabel totalLbl

  Label that displays the total amount.
- javax.swing.JLabel vatLbl

  Label that displays the VAT amount.

Static Private Attributes

- static final String PRIMARY_FONT = "Yu Gothic UI"

  Primary font used in the GUI.
- static final String PRIMARY_FONT_LIGHT = "Yu Gothic UI Light"
- static final Logger LOGGER = LoggerFactory.getLogger(Order.class)

  Logger for this class.

9.33.1 Detailed Description

Main menu window for taking customer orders.

```
Swing window that allows the cashier to:
- select quantities of drinks, appetizers and main courses.
- see the items and prices loaded dynamically from the database.
- calculate subtotal, VAT and total.
- save a simple text receipt to disk.

All menu items and prices are now loaded from the backend
using ProductService (instead of being hardcoded).
```

Definition at line 31 of file Order.java.

9.33.2 Constructor & Destructor Documentation

9.33.2.1 Order()

```
es.ull.esit.app.Order.Order ()  [inline]
```

Constructor.

```
Creates the order menu window, initializes all Swing components,
configures the table models and loads the menu from the backend.
```

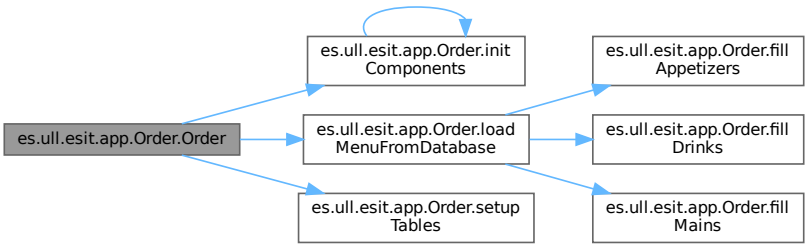
Definition at line 77 of file Order.java.

```
00077                 {
00078     initComponents();
00079
00080     // Initialize the Service Layer.
00081     ApiClient client = new ApiClient("http://localhost:8080");
00082     this.productService = new ProductService(client);
00083
00084     // Initialize receipt number label.
00085     receiptNoLbl.setText("Receipt No. : " + receiptNo);
00086
00087     // Configure tables and models.
00088     setupTables();
00089
00090     // Load menu items from database via REST API.
00091     loadMenuFromDatabase();
00092   }
```

References initComponents(), loadMenuFromDatabase(), receiptNoLbl, and setupTables().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

9.33.3 Member Function Documentation

9.33.3.1 fillAppetizers()

```
void es.ull.esit.app.Order.fillAppetizers (
            java.util.List< Appetizer > appetizers)  [inline], [private]
```

Fills the appetizers table model with data from the backend.

Parameters

appetizers	[java.util.List<Appetizer>] List of appetizers retrieved from the backend.

Definition at line 243 of file Order.java.

```
00243                                                                    {
00244     appetizersModel.setRowCount(0);
00245     for (Appetizer a : appetizers) {
00246       appetizersModel.addRow(new Object[] {
00247           a.getAppetizersId(),
00248           a.getItemAppetizers(),
00249           a.getAppetizersPrice(),
00250           0
00251       });
00252     }
00253   }
```

References appetizersModel.

Referenced by loadMenuFromDatabase().

Here is the caller graph for this function:

9.33.3.2 fillDrinks()

```
void es.ull.esit.app.Order.fillDrinks (
            java.util.List< Drink > drinks)  [inline], [private]
```

Fills the drinks table model with data from the backend.

Parameters

drinks	[java.util.List<Drink>] List of drinks retrieved from the backend.

Definition at line 226 of file Order.java.

```
00226                                                                {
00227     drinksModel.setRowCount(0);
00228     for (Drink d : drinks) {
00229       drinksModel.addRow(new Object[] {
00230           d.getDrinksId(),
00231           d.getItemDrinks(),
```

```
00232           d.getDrinksPrice(),
00233           0 // initial quantity
00234       });
00235     }
00236   }
```

References drinksModel.

Referenced by loadMenuFromDatabase().

Here is the caller graph for this function:

9.33.3.3 fillMains()

```
void es.ull.esit.app.Order.fillMains (
            java.util.List< MainCourse > mains)  [inline], [private]
```

Fills the main courses table model with data from the backend.

Parameters

mains	[java.util.List<MainCourse>] List of main courses retrieved from the backend.

Definition at line 260 of file Order.java.

```
00260                                                       {
00261     mainsModel.setRowCount(0);
00262     for (MainCourse m : mains) {
00263       mainsModel.addRow(new Object[] {
00264           m.getFoodId(),
00265           m.getItemFood(),
00266           m.getFoodPrice(),
00267           0
00268       });
00269     }
00270   }
```

References mainsModel.

Referenced by loadMenuFromDatabase().

Here is the caller graph for this function:

9.33.3.4 goBackMenueBtnActionPerformed()


```
void es.ull.esit.app.Order.goBackMenueBtnActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Returns to the Login window.

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by clicking the "Go Back" button.

Definition at line 749 of file Order.java.

```
00749                                                                    {//
      GEN-FIRST:event_goBackMenueBtnActionPerformed
00750     this.dispose();
00751     new Login().setVisible(true);
00752   }// GEN-LAST:event_goBackMenueBtnActionPerformed
```

9.33.3.5 initComponents()

```
void es.ull.esit.app.Order.initComponents ()  [inline], [private]
```

Initializes GUI components.

```
Generated by the Form Editor: do not modify manually.
Creates and arranges the panels for Drinks, Appetizers,
Main courses and the Receipt summary, as well as all labels,
tables and buttons.
```

Definition at line 339 of file Order.java.

```
00339                               {
00340
00341     jPanel2 = new javax.swing.JPanel();
00342     drinksPnl = new javax.swing.JPanel();
00343     drinksScroll = new javax.swing.JScrollPane();
00344     drinksTable = new javax.swing.JTable();
00345     appetizerPnl = new javax.swing.JPanel();
00346     appetizersScroll = new javax.swing.JScrollPane();
00347     appetizersTable = new javax.swing.JTable();
00348     mainCoursePnl = new javax.swing.JPanel();
00349     mainsScroll = new javax.swing.JScrollPane();
00350     mainsTable = new javax.swing.JTable();
00351     jPanel1 = new javax.swing.JPanel();
00352     subTotalLbl = new javax.swing.JLabel();
00353     vatLbl = new javax.swing.JLabel();
00354     totalLbl = new javax.swing.JLabel();
00355     receiptNoLbl = new javax.swing.JLabel();
00356     payBtn = new javax.swing.JButton();
00357     newReceiptBtn = new javax.swing.JButton();
00358     saveReceiptBtn = new javax.swing.JButton();
00359     jLabel1 = new javax.swing.JLabel();
00360     goBackMenueBtn = new javax.swing.JButton();
00361     jLabel2 = new javax.swing.JLabel();
00362
00363     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00364     setTitle("Menu");
00365     setBackground(new java.awt.Color(204, 204, 204));
00366     setResizable(false);
00367
00368     jPanel2.setBackground(new java.awt.Color(248, 244, 230));
00369
00370     drinksPnl.setBackground(new java.awt.Color(248, 244, 230));
00371     drinksPnl.setBorder(javax.swing.BorderFactory.createTitledBorder(
00372         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Drinks",
00373         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00374         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00375     drinksPnl.setPreferredSize(new java.awt.Dimension(260, 278));
00376
00377     drinksTable.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00378     drinksTable.setModel(new javax.swing.table.DefaultTableModel(
00379         new Object[][] {
00380
00381         },
00382         new String[] {
00383
00384         }));
00385     drinksTable.getTableHeader().setReorderingAllowed(false);
00386     drinksScroll.setViewportView(drinksTable);
00387
```

```
00388     javax.swing.GroupLayout drinksPnlLayout = new javax.swing.GroupLayout(drinksPnl);
00389     drinksPnl.setLayout(drinksPnlLayout);
00390     drinksPnlLayout.setHorizontalGroup(
00391         drinksPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00392             .addGroup(drinksPnlLayout.createSequentialGroup()
00393                 .addContainerGap()
00394                 .addComponent(drinksScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 281,
      Short.MAX_VALUE)
00395                 .addContainerGap()));
00396     drinksPnlLayout.setVerticalGroup(
00397         drinksPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00398             .addGroup(drinksPnlLayout.createSequentialGroup()
00399                 .addContainerGap()
00400                 .addComponent(drinksScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 186,
      Short.MAX_VALUE)
00401                 .addContainerGap()));
00402
00403     appetizerPnl.setBackground(new java.awt.Color(248, 244, 230));
00404     appetizerPnl.setBorder(javax.swing.BorderFactory.createTitledBorder(
00405         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Appetizers",
00406         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00407         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00408     appetizerPnl.setPreferredSize(new java.awt.Dimension(260, 278));
00409
00410     appetizersTable.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00411     appetizersTable.setModel(new javax.swing.table.DefaultTableModel(
00412         new Object[][] {
00413
00414         },
00415         new String[] {
00416
00417         }));
00418     appetizersTable.getTableHeader().setReorderingAllowed(false);
00419     appetizersScroll.setViewportView(appetizersTable);
00420
00421     javax.swing.GroupLayout appetizerPnlLayout = new javax.swing.GroupLayout(appetizerPnl);
00422     appetizerPnl.setLayout(appetizerPnlLayout);
00423     appetizerPnlLayout.setHorizontalGroup(
00424         appetizerPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00425             .addGroup(appetizerPnlLayout.createSequentialGroup()
00426                 .addContainerGap()
00427                 .addComponent(appetizersScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 368,
      Short.MAX_VALUE)
00428                 .addContainerGap()));
00429     appetizerPnlLayout.setVerticalGroup(
00430         appetizerPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00431             .addGroup(appetizerPnlLayout.createSequentialGroup()
00432                 .addContainerGap()
00433                 .addComponent(appetizersScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 186,
      Short.MAX_VALUE)
00434                 .addContainerGap()));
00435
00436     mainCoursePnl.setBackground(new java.awt.Color(248, 244, 230));
00437     mainCoursePnl.setBorder(javax.swing.BorderFactory.createTitledBorder(
00438         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Main Course",
00439         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00440         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00441     mainCoursePnl.setPreferredSize(new java.awt.Dimension(260, 278));
00442
00443     mainsTable.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00444     mainsTable.setModel(new javax.swing.table.DefaultTableModel(
00445         new Object[][] {
00446
00447         },
00448         new String[] {
00449
00450         }));
00451     mainsTable.getTableHeader().setReorderingAllowed(false);
00452     mainsScroll.setViewportView(mainsTable);
00453
00454     javax.swing.GroupLayout mainCoursePnlLayout = new javax.swing.GroupLayout(mainCoursePnl);
00455     mainCoursePnl.setLayout(mainCoursePnlLayout);
00456     mainCoursePnlLayout.setHorizontalGroup(
00457         mainCoursePnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00458             .addGroup(mainCoursePnlLayout.createSequentialGroup()
00459                 .addContainerGap()
00460                 .addComponent(mainsScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 427, Short.MAX_VALUE)
00461                 .addContainerGap()));
00462     mainCoursePnlLayout.setVerticalGroup(
00463         mainCoursePnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00464             .addGroup(mainCoursePnlLayout.createSequentialGroup()
00465                 .addContainerGap()
00466                 .addComponent(mainsScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 186, Short.MAX_VALUE)
00467                 .addContainerGap()));
00468
00469     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00470     jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(
```

```
00471         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Receipt",
00472         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00473         new java.awt.Font(PRIMARY_FONT, 0, 14))); // NOI18N
00474 
00475     subTotalLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 18)); // NOI18N
00476     subTotalLbl.setText("SubTotal: 0.0 SR");
00477 
00478     vatLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 18)); // NOI18N
00479     vatLbl.setText("VAT included: 0.0 SR");
00480 
00481     totalLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 18)); // NOI18N
00482     totalLbl.setText("Total: 0.0 SR");
00483 
00484     receiptNoLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 14)); // NOI18N
00485     receiptNoLbl.setText("Receipt No. : 0");
00486 
00487     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00488     jPanel1.setLayout(jPanel1Layout);
00489     jPanel1Layout.setHorizontalGroup(
00490         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00491             .addGroup(jPanel1Layout.createSequentialGroup()
00492                 .addContainerGap()
00493                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00494                     .addComponent(subTotalLbl)
00495                     .addComponent(vatLbl)
00496                     .addComponent(totalLbl)
00497                     .addComponent(receiptNoLbl))
00498                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)));
00499     jPanel1Layout.setVerticalGroup(
00500         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00501             .addGroup(jPanel1Layout.createSequentialGroup()
00502                 .addContainerGap()
00503                 .addComponent(subTotalLbl)
00504                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00505                 .addComponent(vatLbl)
00506                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00507                 .addComponent(totalLbl)
00508                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 10,
      Short.MAX_VALUE)
00509                 .addComponent(receiptNoLbl)));
00510 
00511     payBtn.setFont(new java.awt.Font("Lucida Grande", 1, 18)); // NOI18N
00512     payBtn.setText("Pay");
00513     payBtn.addActionListener(this::payBtnActionPerformed);
00514 
00515     newReceiptBtn.setFont(new java.awt.Font(PRIMARY_FONT, 1, 14)); // NOI18N
00516     newReceiptBtn.setText("New Receipt");
00517     newReceiptBtn.addActionListener(this::newReceiptBtnActionPerformed);
00518 
00519     saveReceiptBtn.setFont(new java.awt.Font(PRIMARY_FONT, 1, 14)); // NOI18N
00520     saveReceiptBtn.setText("Save Receipt");
00521     saveReceiptBtn.addActionListener(this::saveReceiptBtnActionPerformed);
00522 
00523     jLabel1.setBackground(new java.awt.Color(255, 153, 0));
00524     jLabel1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 36)); // NOI18N
00525     jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00526     jLabel1.setText("BLACK PLATE MENU");
00527 
00528     goBackMenueBtn.setFont(new java.awt.Font(PRIMARY_FONT, 1, 14)); // NOI18N
00529     goBackMenueBtn.setText("Go Back");
00530     goBackMenueBtn.addActionListener(this::goBackMenueBtnActionPerformed);
00531 
00532     jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
      NOI18N
00533 
00534     javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
00535     jPanel2.setLayout(jPanel2Layout);
00536     jPanel2Layout.setHorizontalGroup(
00537         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00538             .addGroup(jPanel2Layout.createSequentialGroup()
00539                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00540                     .addGroup(jPanel2Layout.createSequentialGroup()
00541                         .addGap(40, 40, 40)
00542                         .addComponent(drinksPnl, javax.swing.GroupLayout.PREFERRED_SIZE, 305,
00543                             javax.swing.GroupLayout.PREFERRED_SIZE)
00544                         .addGap(18, 18, 18)
00545                         .addComponent(appetizerPnl, javax.swing.GroupLayout.PREFERRED_SIZE, 392,
00546                             javax.swing.GroupLayout.PREFERRED_SIZE)
00547                         .addGap(18, 18, 18)
00548                         .addComponent(mainCoursePnl, javax.swing.GroupLayout.PREFERRED_SIZE, 451,
00549                             javax.swing.GroupLayout.PREFERRED_SIZE))
00550                     .addGroup(jPanel2Layout.createSequentialGroup()
00551                         .addGap(251, 251, 251)
00552                         .addComponent(jLabel2)
00553                         .addGap(18, 18, 18)
00554                         .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 433,
00555                             javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
00556                     .addGroup(jPanel2Layout.createSequentialGroup()
00557                         .addGap(16, 16, 16)
00558                         .addComponent(goBackMenueBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 130,
00559                             javax.swing.GroupLayout.PREFERRED_SIZE)
00560                         .addGap(147, 147, 147)
00561                         .addComponent(payBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 180,
00562                             javax.swing.GroupLayout.PREFERRED_SIZE)
00563                         .addGap(18, 18, 18)
00564
    .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00565                         .addComponent(newReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 220,
00566                             javax.swing.GroupLayout.PREFERRED_SIZE)
00567                         .addComponent(saveReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 220,
00568                             javax.swing.GroupLayout.PREFERRED_SIZE))
00569                         .addGap(28, 28, 28)
00570                         .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
00571                             javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)))
00572                 .addContainerGap(30, Short.MAX_VALUE)));
00573     jPanel2Layout.setVerticalGroup(
00574         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00575             .addGroup(jPanel2Layout.createSequentialGroup()
00576                 .addContainerGap(16, Short.MAX_VALUE)
00577                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00578                     .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.TRAILING)
00579                     .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING,
00580                         javax.swing.GroupLayout.PREFERRED_SIZE, 60,
    javax.swing.GroupLayout.PREFERRED_SIZE))
00581                 .addGap(40, 40, 40)
00582                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
    false)
00583                     .addComponent(appetizerPnl, javax.swing.GroupLayout.DEFAULT_SIZE, 230,
    Short.MAX_VALUE)
00584                     .addComponent(mainCoursePnl, javax.swing.GroupLayout.DEFAULT_SIZE, 230,
    Short.MAX_VALUE)
00585                     .addComponent(drinksPnl, javax.swing.GroupLayout.DEFAULT_SIZE, 230,
    Short.MAX_VALUE))
00586                 .addGap(18, 18, 18)
00587                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00588                     .addComponent(goBackMenueBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
00589                         javax.swing.GroupLayout.PREFERRED_SIZE)
00590                     .addGroup(jPanel2Layout.createSequentialGroup()
00591                         .addComponent(newReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
00592                             javax.swing.GroupLayout.PREFERRED_SIZE)
00593                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00594                         .addComponent(saveReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
00595                             javax.swing.GroupLayout.PREFERRED_SIZE))
00596                     .addComponent(payBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
00597                         javax.swing.GroupLayout.PREFERRED_SIZE)
00598                     .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
00599                         javax.swing.GroupLayout.PREFERRED_SIZE))
00600                 .addGap(30, 30, 30)));
00601
00602     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00603     getContentPane().setLayout(layout);
00604     layout.setHorizontalGroup(
00605         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00606             .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
00607                 Short.MAX_VALUE));
00608     layout.setVerticalGroup(
00609         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00610             .addGroup(layout.createSequentialGroup()
00611                 .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
00612                     Short.MAX_VALUE)
00613                 .addContainerGap()));
00614
00615     pack();
00616     setLocationRelativeTo(null);
00617   }// </editor-fold>//GEN-END:initComponents
```

References appetizerPnl, appetizersScroll, appetizersTable, drinksPnl, drinksScroll, drinksTable, goBackMenueBtn, initComponents(), jLabel1, jLabel2, jPanel1, jPanel2, mainCoursePnl, mainsScroll, mainsTable, newReceiptBtn, payBtn, PRIMARY_FONT, PRIMARY_FONT_LIGHT, receiptNoLbl, saveReceiptBtn, subTotalLbl, totalLbl, and vatLbl.

Referenced by initComponents(), and Order().

Here is the call graph for this function:

Here is the caller graph for this function:

9.33.3.6 loadMenuFromDatabase()

```
void es.ull.esit.app.Order.loadMenuFromDatabase ()  [inline], [private]
```

Loads menu data (drinks, appetizers, main courses) from the backend in a background thread.

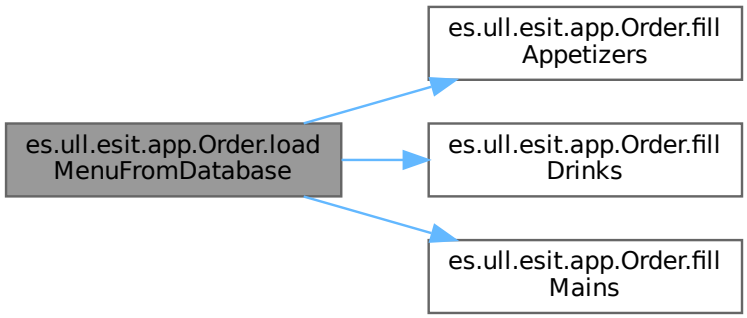
Definition at line 197 of file Order.java.

```
00197                                     {
00198     new Thread(() -> {
00199       try {
00200         java.util.List<Drink> drinks = productService.getAllDrinks();
00201         java.util.List<Appetizer> appetizers = productService.getAllAppetizers();
00202         java.util.List<MainCourse> mains = productService.getAllMainCourses();
00203 
00204         SwingUtilities.invokeLater(() -> {
00205           fillDrinks(drinks);
00206           fillAppetizers(appetizers);
00207           fillMains(mains);
00208         });
00209 
00210     } catch (Exception ex) {
00211       LOGGER.error("Error loading menu from backend", ex);
00212       SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog(
00213       this,
00214       "Error loading menu from backend:\n" + ex.getMessage(),
00215       "Menu loading error",
00216       JOptionPane.ERROR_MESSAGE));
00217     }
00218       }).start();
00219     }
```

References fillAppetizers(), fillDrinks(), fillMains(), LOGGER, and productService.

Referenced by Order().

Here is the call graph for this function:



Here is the caller graph for this function:

9.33.3.7 main()

```
void es.ull.esit.app.Order.main (
            String[] args)  [inline], [static]
```

Standalone entry point for testing the Order window.

```
Sets the Nimbus look and feel if available and displays a new
instance of Order. In the normal application flow, this window
is opened after a successful login.
```

Parameters

args	[String[]] Command line arguments (not used).

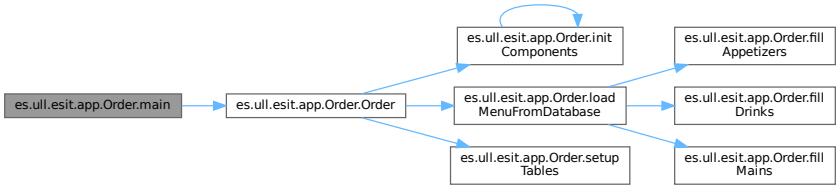
Definition at line 763 of file Order.java.

```
00763                                     {
00764     /* Set the Nimbus look and feel */
00765     // <editor-fold defaultstate="collapsed" desc=" Look and feel setting code
00766     // (optional) ">
00767     /*
00768      * If Nimbus (introduced in Java SE 6) is not available, stay with the default
00769      * look and feel.
00770      * For details see
00771      * http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
00772      */
```

```
00773     try {
00774       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00775         if ("Nimbus".equals(info.getName())) {
00776           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00777           break;
00778         }
00779       }
00780     } catch (ClassNotFoundException | javax.swing.UnsupportedLookAndFeelException |
00781         InstantiationException | IllegalAccessException ex) {
00782       java.util.logging.Logger.getLogger(Order.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00783     }
00784     // </editor-fold>
00785
00786     /* Create and display the form */
00787     java.awt.EventQueue.invokeLater(() -> new Order().setVisible(true));
00788   }
```

References Order().

Here is the call graph for this function:



9.33.3.8 newReceiptBtnActionPerformed()

```
void es.ull.esit.app.Order.newReceiptBtnActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Starts a new receipt and resets the form to its initial state.

```
Only works when the current total is not zero. If so, it:
- resets Qty column to 0 in all tables,
- resets subtotal, VAT and total labels,
- clears internal subtotal, VAT and total variables,
- clears lastBill,
- increments the receipt number and updates its label.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by clicking the "New Receipt" button.

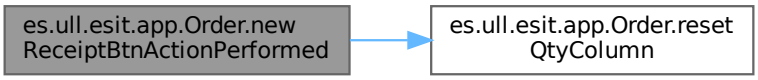
Definition at line 717 of file Order.java.

```
00717                                                                    {//
      GEN-FIRST:event_newReceiptBtnActionPerformed
00718     if (total == 0.0) {
00719       // Nothing to reset
00720       return;
00721     }
00722
00723     resetQtyColumn(drinksModel);
00724     resetQtyColumn(appetizersModel);
00725     resetQtyColumn(mainsModel);
```

```
00726
00727     subTotal = 0.0;
00728     vat = 0.0;
00729     total = 0.0;
00730     lastBill = null;
00731
00732     subTotalLbl.setText("SubTotal: 0.0 SR");
00733     vatLbl.setText("VAT included: 0.0 SR");
00734     totalLbl.setText("Total: 0.0 SR");
00735
00736     receiptNo++;
00737     receiptNoLbl.setText("Receipt No. : " + receiptNo);
00738   }// GEN-LAST:event_newReceiptBtnActionPerformed
```

References appetizersModel, drinksModel, lastBill, mainsModel, receiptNoLbl, resetQtyColumn(), subTotalLbl, totalLbl, and vatLbl.

Here is the call graph for this function:



9.33.3.9 payBtnActionPerformed()

```
void es.ull.esit.app.Order.payBtnActionPerformed (
             java.awt.event.ActionEvent evt)  [inline], [private]
```

Calculates subtotal, VAT and total and shows a confirmation dialog.

```
   Steps:
   - Sums all line prices from the three tables.
   - Uses OrderService to compute BillResult (subtotal, VAT, total).
   - Updates the labels in the Receipt panel.
   - If no items were selected (sum == 0), shows an information dialog.
   - Otherwise, shows a "Paid successfully" dialog.
```

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by clicking the "Pay" button.

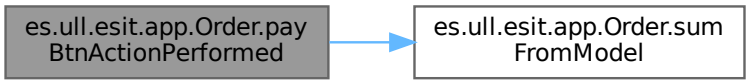
Definition at line 632 of file Order.java.

```
00632                                                                {//
      GEN-FIRST:event_payBtnActionPerformed
00633     double itemsTotal = 0.0;
00634
00635     itemsTotal += sumFromModel(drinksModel);
00636     itemsTotal += sumFromModel(appetizersModel);
00637     itemsTotal += sumFromModel(mainsModel);
00638
00639     if (itemsTotal == 0.0) {
00640       JOptionPane.showMessageDialog(
00641           this,
00642           "Please select at least one item (Qty > 0).",
00643           "No items selected",
00644           JOptionPane.INFORMATION_MESSAGE);
```

```
00645       return;
00646     }
00647
00648     // Calculate bill using the dedicated service
00649     lastBill = orderService.calculateBill(itemsTotal);
00650
00651     subTotal = lastBill.getSubTotal();
00652     vat = lastBill.getVat();
00653     total = lastBill.getTotal();
00654
00655     subTotalLbl.setText("SubTotal: " + subTotal + " SR");
00656     vatLbl.setText("VAT included: " + vat + " SR");
00657     totalLbl.setText("Total: " + total + " SR");
00658
00659     JOptionPane.showMessageDialog(this, "Paid successfully");
00660   }// GEN-LAST:event_payBtnActionPerformed
```

References appetizersModel, drinksModel, lastBill, mainsModel, orderService, subTotalLbl, sumFromModel(), totalLbl, and vatLbl.

Here is the call graph for this function:



9.33.3.10 resetQtyColumn()

```
void es.ull.esit.app.Order.resetQtyColumn (
            DefaultTableModel model)  [inline], [private]
```

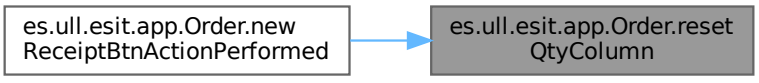
Sets the Qty column of the given model to 0 in all rows.

Definition at line 312 of file Order.java.

```
00312                                                {
00313     for (int row = 0; row < model.getRowCount(); row++) {
00314       model.setValueAt(0, row, 3); // column 3 is Qty
00315     }
00316   }
```

Referenced by newReceiptBtnActionPerformed().

Here is the caller graph for this function:



9.33.3.11 saveReceiptBtnActionPerformed()

```
void es.ull.esit.app.Order.saveReceiptBtnActionPerformed (
            java.awt.event.ActionEvent evt)  [inline], [private]
```

Saves the current bill information to a text file through OrderService.

The file is created under the "receipts" directory with the name:

- "billNo.<receiptNo>.txt"

If there is no calculated bill (lastBill == null or total == 0), an information dialog is shown and no file is created.

Parameters

evt	[java.awt.event.ActionEvent] Event triggered by clicking the "Save Receipt" button.

Definition at line 675 of file Order.java.

```
00675                                                                    {//
      GEN-FIRST:event_saveReceiptBtnActionPerformed
00676     try {
00677       if (lastBill == null || lastBill.getTotal() == 0.0) {
00678         JOptionPane.showMessageDialog(
00679             this,
00680             "There is no paid bill to save.\nPlease press Pay first.",
00681             "No bill",
00682             JOptionPane.INFORMATION_MESSAGE);
00683         return;
00684       }
00685
00686       orderService.generateReceiptFile(receiptNo, lastBill);
00687
00688       JOptionPane.showMessageDialog(
00689           this,
00690           "Receipt number: " + receiptNo + " has been saved successfully.",
00691           "Receipt saved",
00692           JOptionPane.INFORMATION_MESSAGE);
00693
00694     } catch (Exception ex) {
00695       LOGGER.error("Error saving receipt", ex);
00696       JOptionPane.showMessageDialog(
00697           this,
00698           "Error saving receipt:\n" + ex.getMessage(),
00699           "Error",
00700           JOptionPane.ERROR_MESSAGE);
00701     }
00702   }// GEN-LAST:event_saveReceiptBtnActionPerformed
```

References lastBill, LOGGER, and orderService.

9.33.3.12 setupTables()

```
void es.ull.esit.app.Order.setupTables ()  [inline], [private]
```

Configures the three tables (Drinks, Appetizers, Main Course).

```
   Each table has four columns:
   - ID (not editable)
   - Item (not editable)
   - Price (not editable)
   - Qty (editable, quantity selected by the cashier)
```

Definition at line 133 of file Order.java.

```
00133                             {
00134     // DRINKS
00135     drinksModel = new DefaultTableModel(
00136         new Object[] { "ID", "Item", "Price (SR)", "Qty" }, 0) {
00137       @Override
00138       public boolean isCellEditable(int row, int column) {
00139         // Only Qty column is editable
00140         return column == 3;
00141       }
00142
00143       @Override
00144       public Class<?> getColumnClass(int columnIndex) {
00145         return switch (columnIndex) {
00146           case 0 -> Long.class; // ID
00147           case 2, 3 -> Integer.class; // Price, Qty
00148           default -> String.class;
00149         };
00150       }
00151     };
00152     drinksTable.setModel(drinksModel);
00153
00154     // APPETIZERS
00155     appetizersModel = new DefaultTableModel(
00156         new Object[] { "ID", "Item", "Price (SR)", "Qty" }, 0) {
00157       @Override
00158       public boolean isCellEditable(int row, int column) {
00159         return column == 3;
00160       }
00161
00162       @Override
00163       public Class<?> getColumnClass(int columnIndex) {
00164         return switch (columnIndex) {
00165           case 0 -> Long.class;
00166           case 2, 3 -> Integer.class;
00167           default -> String.class;
00168         };
00169       }
00170     };
00171     appetizersTable.setModel(appetizersModel);
00172
00173     // MAIN COURSES
00174     mainsModel = new DefaultTableModel(
00175         new Object[] { "ID", "Item", "Price (SR)", "Qty" }, 0) {
00176       @Override
00177       public boolean isCellEditable(int row, int column) {
00178         return column == 3;
00179       }
00180
00181       @Override
00182       public Class<?> getColumnClass(int columnIndex) {
00183         return switch (columnIndex) {
00184           case 0 -> Long.class;
00185           case 2, 3 -> Integer.class;
00186           default -> String.class;
00187         };
00188       }
00189     };
00190     mainsTable.setModel(mainsModel);
00191   }
```

References appetizersModel, appetizersTable, drinksModel, drinksTable, mainsModel, and mainsTable.

Referenced by Order().

Here is the caller graph for this function:

9.33.3.13 sumFromModel()

```
double es.ull.esit.app.Order.sumFromModel (
            DefaultTableModel model)  [inline], [private]
```

Sums the total price for a given table model.

```
For each row:
- reads Price (column 2) and Qty (column 3),
- if Qty > 0, accumulates (Qty * Price).
```

Parameters

model	[javax.swing.table.DefaultTableModel] Table model (drinksModel, appetizersModel or mainsModel).

Returns

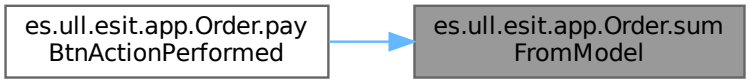
[double] Sum of the line totals in that model.

Definition at line 282 of file Order.java.

```
00282                                                    {
00283     double sum = 0.0;
00284     for (int row = 0; row < model.getRowCount(); row++) {
00285       Object qtyObj = model.getValueAt(row, 3); // Qty
00286       Object priceObj = model.getValueAt(row, 2); // Price
00287
00288       if (qtyObj == null || priceObj == null)
00289         continue;
00290
00291       int qty;
00292       int price;
00293       try {
00294         qty = ((Number) qtyObj).intValue();
00295         price = ((Number) priceObj).intValue();
00296       } catch (ClassCastException e) {
00297         // Fallback in case something comes as String
00298         qty = Integer.parseInt(qtyObj.toString());
00299         price = Integer.parseInt(priceObj.toString());
00300       }
00301
00302       if (qty > 0) {
00303         sum += qty * price;
00304       }
00305     }
00306     return sum;
00307   }
```

Referenced by payBtnActionPerformed().

Here is the caller graph for this function:



9.33.4 Member Data Documentation

9.33.4.1 appetizerPnl

```
javax.swing.JPanel es.ull.esit.app.Order.appetizerPnl  [private]
```

Panel that groups all appetizer items and controls.

Definition at line 794 of file Order.java.

Referenced by initComponents().

9.33.4.2 appetizersModel

```
DefaultTableModel es.ull.esit.app.Order.appetizersModel  [private]
```

Definition at line 49 of file Order.java.

Referenced by fillAppetizers(), newReceiptBtnActionPerformed(), payBtnActionPerformed(), and setupTables().

9.33.4.3 appetizersScroll

```
javax.swing.JScrollPane es.ull.esit.app.Order.appetizersScroll  [private]
```

Scroll pane for appetizers table.

Definition at line 802 of file Order.java.

Referenced by initComponents().

9.33.4.4 appetizersTable

```
javax.swing.JTable es.ull.esit.app.Order.appetizersTable  [private]
```

Table for appetizers items.

Definition at line 800 of file Order.java.

Referenced by initComponents(), and setupTables().

9.33.4.5 drinksModel

```
DefaultTableModel es.ull.esit.app.Order.drinksModel  [private]
```

Table models for the three product categories.

Definition at line 48 of file Order.java.

Referenced by fillDrinks(), newReceiptBtnActionPerformed(), payBtnActionPerformed(), and setupTables().

9.33.4.6 drinksPnl

```
javax.swing.JPanel es.ull.esit.app.Order.drinksPnl  [private]
```

Panel that groups all drink items and controls.

Definition at line 796 of file Order.java.

Referenced by initComponents().

9.33.4.7 drinksScroll

```
javax.swing.JScrollPane es.ull.esit.app.Order.drinksScroll  [private]
```

Scroll pane for drinks table.

Definition at line 806 of file Order.java.

Referenced by initComponents().

9.33.4.8 drinksTable

```
javax.swing.JTable es.ull.esit.app.Order.drinksTable  [private]
```

Table for drinks items.

Definition at line 804 of file Order.java.

Referenced by initComponents(), and setupTables().

9.33.4.9 goBackMenueBtn

```
javax.swing.JButton es.ull.esit.app.Order.goBackMenueBtn  [private]
```

Button to go back to the Login window.

Definition at line 808 of file Order.java.

Referenced by initComponents().

9.33.4.10 jLabel1

```
javax.swing.JLabel es.ull.esit.app.Order.jLabel1  [private]
```

Title label "BLACK PLATE MENU".

Definition at line 810 of file Order.java.

Referenced by initComponents().

9.33.4.11 jLabel2

```
javax.swing.JLabel es.ull.esit.app.Order.jLabel2  [private]
```

Logo / screenshot label on the top left.

Definition at line 812 of file Order.java.

Referenced by initComponents().

9.33.4.12 jPanel1

```
javax.swing.JPanel es.ull.esit.app.Order.jPanel1  [private]
```

Panel that shows subtotal, VAT, total and receipt number.

Definition at line 814 of file Order.java.

Referenced by initComponents().

9.33.4.13 jPanel2

```
javax.swing.JPanel es.ull.esit.app.Order.jPanel2  [private]
```

Main container panel of the Order window.

Definition at line 816 of file Order.java.

Referenced by initComponents().

9.33.4.14 lastBill

```
transient BillResult es.ull.esit.app.Order.lastBill  [private]
```

Last calculated bill (after pressing Pay).

Definition at line 53 of file Order.java.

Referenced by newReceiptBtnActionPerformed(), payBtnActionPerformed(), and saveReceiptBtnActionPerformed().

9.33.4.15 LOGGER

```
final Logger es.ull.esit.app.Order.LOGGER = LoggerFactory.getLogger(Order.class)  [static],
[private]
```

Logger for this class.

Replaces printStackTrace() debug output.

Definition at line 45 of file Order.java.

Referenced by loadMenuFromDatabase(), and saveReceiptBtnActionPerformed().

9.33.4.16 mainCoursePnl

```
javax.swing.JPanel es.ull.esit.app.Order.mainCoursePnl  [private]
```

Panel that groups all main course items and controls.

Definition at line 798 of file Order.java.

Referenced by initComponents().

9.33.4.17 mainsModel

```
DefaultTableModel es.ull.esit.app.Order.mainsModel  [private]
```

Definition at line 50 of file Order.java.

Referenced by fillMains(), newReceiptBtnActionPerformed(), payBtnActionPerformed(), and setupTables().

9.33.4.18 mainsScroll

```
javax.swing.JScrollPane es.ull.esit.app.Order.mainsScroll  [private]
```

Scroll pane for main course table.

Definition at line 820 of file Order.java.

Referenced by initComponents().

9.33.4.19 mainsTable

```
javax.swing.JTable es.ull.esit.app.Order.mainsTable  [private]
```

Table for main course items.

Definition at line 818 of file Order.java.

Referenced by initComponents(), and setupTables().

9.33.4.20 newReceiptBtn

```
javax.swing.JButton es.ull.esit.app.Order.newReceiptBtn  [private]
```

Button to start a new receipt.

Definition at line 822 of file Order.java.

Referenced by initComponents().

9.33.4.21 orderService

```
final transient OrderService es.ull.esit.app.Order.orderService = new OrderService()  [private]
```

Service used to calculate bill totals and generate receipt files.

Definition at line 42 of file Order.java.

Referenced by payBtnActionPerformed(), and saveReceiptBtnActionPerformed().

9.33.4.22 payBtn

```
javax.swing.JButton es.ull.esit.app.Order.payBtn  [private]
```

Button to calculate and confirm payment.

Definition at line 824 of file Order.java.

Referenced by initComponents().

9.33.4.23 PRIMARY_FONT

```
final String es.ull.esit.app.Order.PRIMARY_FONT = "Yu Gothic UI"  [static], [private]
```

Primary font used in the GUI.

Definition at line 34 of file Order.java.

Referenced by initComponents().

9.33.4.24 PRIMARY_FONT_LIGHT

```
final String es.ull.esit.app.Order.PRIMARY_FONT_LIGHT = "Yu Gothic UI Light"  [static], [private]
```

Definition at line 36 of file Order.java.

Referenced by initComponents().

9.33.4.25 productService

```
final transient ProductService es.ull.esit.app.Order.productService  [private]
```

Service used to load products from the backend.

Definition at line 39 of file Order.java.

Referenced by loadMenuFromDatabase().

9.33.4.26 receiptNoLbl

```
javax.swing.JLabel es.ull.esit.app.Order.receiptNoLbl  [private]
```

Label that shows the current receipt number.

Definition at line 826 of file Order.java.

Referenced by initComponents(), newReceiptBtnActionPerformed(), and Order().

9.33.4.27 saveReceiptBtn

```
javax.swing.JButton es.ull.esit.app.Order.saveReceiptBtn  [private]
```

Button to save the current receipt to a file.

Definition at line 828 of file Order.java.

Referenced by initComponents().

9.33.4.28 subTotalLbl

```
javax.swing.JLabel es.ull.esit.app.Order.subTotalLbl  [private]
```

Label that displays the subtotal amount.

Definition at line 830 of file Order.java.

Referenced by initComponents(), newReceiptBtnActionPerformed(), and payBtnActionPerformed().

9.33.4.29 totalLbl

```
javax.swing.JLabel es.ull.esit.app.Order.totalLbl  [private]
```

Label that displays the total amount.

Definition at line 832 of file Order.java.

Referenced by initComponents(), newReceiptBtnActionPerformed(), and payBtnActionPerformed().

9.33.4.30 vatLbl

```
javax.swing.JLabel es.ull.esit.app.Order.vatLbl  [private]
```

Label that displays the VAT amount.

Definition at line 834 of file Order.java.

Referenced by initComponents(), newReceiptBtnActionPerformed(), and payBtnActionPerformed().

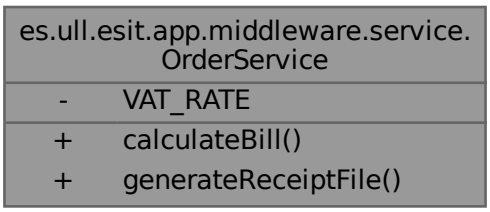
The documentation for this class was generated from the following file:

- Order.java

9.34 es.ull.esit.app.middleware.service.OrderService Class Reference

Service that handles order-related calculations and receipt generation.

Collaboration diagram for es.ull.esit.app.middleware.service.OrderService:



Public Member Functions

- BillResult calculateBill (double itemsTotalSum)

  Calculates subtotal, VAT and total for a given items sum.
- void generateReceiptFile (int receiptNo, BillResult bill) throws java.io.FileNotFoundException

  Generates a local text file representing a receipt.

Static Private Attributes

- static final double VAT_RATE = 0.15

  VAT rate applied to the subtotal (15%).

9.34.1 Detailed Description

Service that handles order-related calculations and receipt generation.

```
Provides methods to compute totals (including VAT) and to generate
local text files representing printed receipts.
```

Definition at line 11 of file OrderService.java.

9.34.2 Member Function Documentation

9.34.2.1 calculateBill()

```
BillResult es.ull.esit.app.middleware.service.OrderService.calculateBill (
            double itemsTotalSum)  [inline]
```

Calculates subtotal, VAT and total for a given items sum.

```
The values are rounded to two decimal places.
```

Parameters

itemsTotalSum	[double] Sum of item prices before VAT.

Returns

[BillResult] Container holding subtotal, VAT and total amounts.

Definition at line 24 of file OrderService.java.

```
00024                                                       {
00025     double vat = itemsTotalSum * VAT_RATE;
00026     double total = itemsTotalSum + vat;
00027
00028     return new BillResult(
00029         Math.round(itemsTotalSum * 100.0) / 100.0,
00030         Math.round(vat * 100.0) / 100.0,
00031         Math.round(total * 100.0) / 100.0);
00032   }
```

References VAT_RATE.

9.34.2.2 generateReceiptFile()

```
void es.ull.esit.app.middleware.service.OrderService.generateReceiptFile (
             int receiptNo,
             BillResult bill) throws java.io.FileNotFoundException  [inline]
```

Generates a local text file representing a receipt.

```
The file is stored under a "receipts" directory created in the
working folder. File name format is "billNo.<receiptNo>.txt".
```

Parameters

receiptNo	[int] Numeric identifier of the receipt.
bill	[BillResult] Calculated bill result to print.

Exceptions

java.io.FileNotFoundException	If the receipt file cannot be created or opened.

Definition at line 44 of file OrderService.java.

```
00044
      {
00045     // Ensure the directory exists.
00046     new java.io.File("receipts").mkdirs();
00047
00048     try (java.io.PrintWriter output = new java.io.PrintWriter("receipts/billNo." + receiptNo +
      ".txt")) {
00049       output.println(" Bill number is: " + receiptNo);
00050       output.println("==============");
00051       output.println("--------------");
00052       output.println("Subtotal is: " + bill.getSubTotal() + " SR");
00053       output.println("vat: " + bill.getVat() + " SR");
00054       output.println("Total is: " + bill.getTotal() + " SR");
00055       output.println();
00056       output.println("THANK YOU FOR ORDERING");
00057     }
00058   }
```

9.34.3 Member Data Documentation

9.34.3.1 VAT_RATE

```
final double es.ull.esit.app.middleware.service.OrderService.VAT_RATE = 0.15  [static], [private]
```

VAT rate applied to the subtotal (15%).

Definition at line 14 of file OrderService.java.

Referenced by calculateBill().

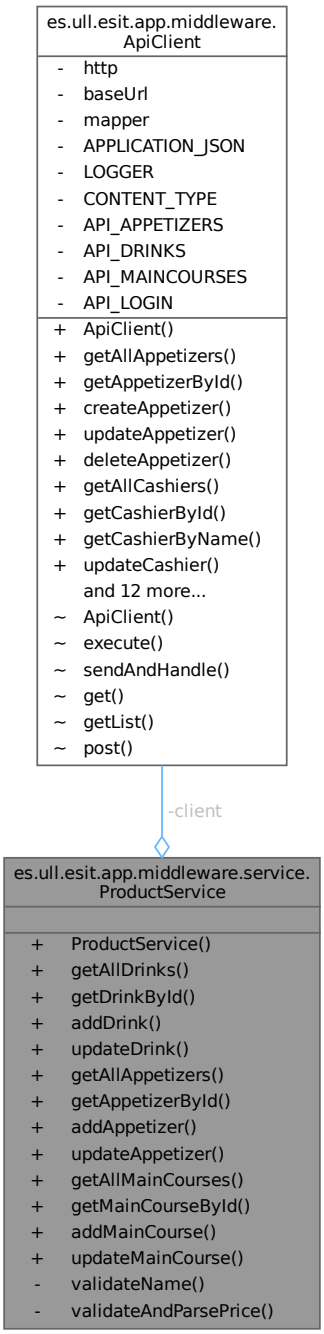
The documentation for this class was generated from the following file:

- OrderService.java

9.35 es.ull.esit.app.middleware.service.ProductService Class Reference

Service handling business logic for menu products.

Collaboration diagram for es.ull.esit.app.middleware.service.ProductService:



Public Member Functions

- ProductService (ApiClient client)

  Constructs a ProductService with the given API client.
- List< Drink > getAllDrinks ()

  Retrieves all drinks from the backend.
- Drink getDrinkById (Long id)

Retrieves a single drink by its identifier.

- void addDrink (String name, String priceStr)

  Adds a new drink to the backend menu.
- void updateDrink (Long id, String name, String priceStr)

  Updates an existing drink in the backend menu.
- List< Appetizer > getAllAppetizers ()

  Retrieves all appetizers from the backend.
- Appetizer getAppetizerById (Long id)

  Retrieves a single appetizer by its identifier.
- void addAppetizer (String name, String priceStr)

  Adds a new appetizer to the backend menu.
- void updateAppetizer (Long id, String name, String priceStr)

  Updates an existing appetizer in the backend menu.
- List< MainCourse > getAllMainCourses ()

  Retrieves all main courses from the backend.
- MainCourse getMainCourseById (Long id)

  Retrieves a single main course by its identifier.
- void addMainCourse (String name, String priceStr)

  Adds a new main course to the backend menu.
- void updateMainCourse (Long id, String name, String priceStr)

  Updates an existing main course in the backend menu.

Private Member Functions

- void validateName (String name)

  Validates that the item name is not null or empty.
- int validateAndParsePrice (String priceStr)

  Validates and parses the price from string to integer.

Private Attributes

- final ApiClient client

  REST API client used to communicate with the backend menu endpoints.

9.35.1 Detailed Description

Service handling business logic for menu products.

```
Decouples Swing UI components from direct REST API calls. It centralizes
validation and orchestrates requests for drinks, appetizers and main
courses.
```

Definition at line 16 of file ProductService.java.

9.35.2 Constructor & Destructor Documentation

9.35.2.1 ProductService()


```
es.ull.esit.app.middleware.service.ProductService.ProductService (
            ApiClient client)  [inline]
```

Parameters

client	[ApiClient] REST client used to perform HTTP requests.

Definition at line 26 of file ProductService.java.

```
00026                                           {
00027     this.client = client;
00028   }
```

References client.

9.35.3 Member Function Documentation

9.35.3.1 addAppetizer()

```
void es.ull.esit.app.middleware.service.ProductService.addAppetizer (
            String name,
            String priceStr)  [inline]
```

Adds a new appetizer to the backend menu.

Validates user input and sends a POST request using the ApiClient.

Parameters

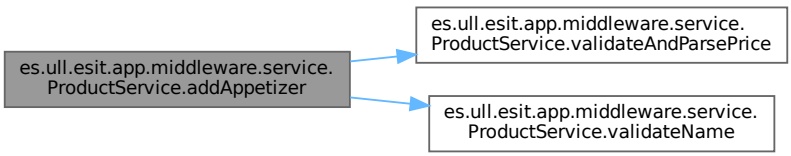
name	[String] Name of the new appetizer.
priceStr	[String] Price entered as a string.

Definition at line 115 of file ProductService.java.

```
00115                                                   {
00116     int price = validateAndParsePrice(priceStr);
00117     validateName(name);
00118
00119     Appetizer newAppetizer = new Appetizer(null, name, price, null);
00120     client.createAppetizer(newAppetizer);
00121   }
```

References client, validateAndParsePrice(), and validateName().

Here is the call graph for this function:



9.35.3.2 addDrink()

```
void es.ull.esit.app.middleware.service.ProductService.addDrink (
            String name,
            String priceStr)  [inline]
```

Adds a new drink to the backend menu.

Validates the user input and sends a POST request using the ApiClient.

Parameters

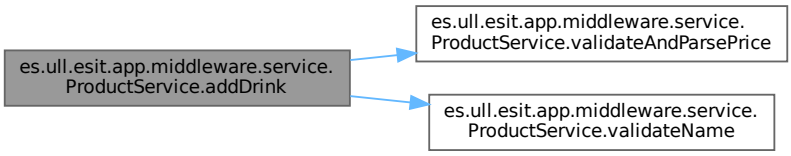
name	[String] Name of the new drink.
priceStr	[String] Price entered as a string.

Definition at line 59 of file ProductService.java.

```
00059                                                {
00060     int price = validateAndParsePrice(priceStr);
00061     validateName(name);
00062
00063     Drink newDrink = new Drink(null, name, price, null);
00064     client.createDrink(newDrink);
00065   }
```

References client, validateAndParsePrice(), and validateName().

Here is the call graph for this function:



9.35.3.3 addMainCourse()

```
void es.ull.esit.app.middleware.service.ProductService.addMainCourse (
            String name,
            String priceStr)  [inline]
```

Adds a new main course to the backend menu.

Validates user input and sends a POST request using the ApiClient.

Parameters

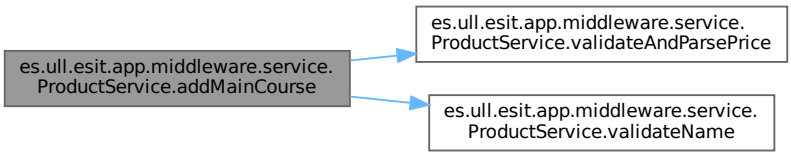
name	[String] Name of the new main course.
priceStr	[String] Price entered as a string.

Definition at line 171 of file ProductService.java.

```
00171                                                       {
00172     int price = validateAndParsePrice(priceStr);
00173     validateName(name);
00174
00175     MainCourse newMainCourse = new MainCourse(null, name, price, null);
00176     client.createMainCourse(newMainCourse);
00177   }
```

References client, validateAndParsePrice(), and validateName().

Here is the call graph for this function:



9.35.3.4 getAllAppetizers()

```
List< Appetizer > es.ull.esit.app.middleware.service.ProductService.getAllAppetizers ()  [inline]
```

Retrieves all appetizers from the backend.

Returns

[List<Appetizer>] List of all appetizers in the menu.

Definition at line 93 of file ProductService.java.

```
00093                                          {
00094     return client.getAllAppetizers();
00095   }
```

References client.

9.35.3.5 getAllDrinks()

```
List< Drink > es.ull.esit.app.middleware.service.ProductService.getAllDrinks ()  [inline]
```

Retrieves all drinks from the backend.

Returns

[List<Drink>] List of all drinks available in the menu.

Definition at line 37 of file ProductService.java.

```
00037                                  {
00038     return client.getAllDrinks();
00039   }
```

References client.

9.35.3.6 getAllMainCourses()

```
List< MainCourse > es.ull.esit.app.middleware.service.ProductService.getAllMainCourses ()
[inline]
```

Retrieves all main courses from the backend.

Returns

[List<MainCourse>] List of all main courses in the menu.

Definition at line 149 of file ProductService.java.

```
00149                                           {
00150     return client.getAllMainCourses();
00151   }
```

References client.

9.35.3.7 getAppetizerById()

```
Appetizer es.ull.esit.app.middleware.service.ProductService.getAppetizerById (
            Long id)  [inline]
```

Retrieves a single appetizer by its identifier.

Parameters

id	[Long] Unique identifier of the appetizer.

Returns

[Appetizer] Appetizer object corresponding to the given id.

Definition at line 103 of file ProductService.java.

```
00103                                               {
00104     return client.getAppetizerById(id);
00105   }
```

References client.

9.35.3.8 getDrinkById()

```
Drink es.ull.esit.app.middleware.service.ProductService.getDrinkById (
            Long id)  [inline]
```

Retrieves a single drink by its identifier.

Parameters

id	[Long] Unique identifier of the drink.

Returns

[Drink] Drink object corresponding to the given id.

Definition at line 47 of file ProductService.java.

```
00047                                       {
00048     return client.getDrinkById(id);
00049   }
```

References client.

9.35.3.9 getMainCourseById()

```
MainCourse es.ull.esit.app.middleware.service.ProductService.getMainCourseById (
            Long id)  [inline]
```

Retrieves a single main course by its identifier.

Parameters

id	[Long] Unique identifier of the main course.

Returns

[MainCourse] MainCourse object corresponding to the given id.

Definition at line 159 of file ProductService.java.

```
00159                                                {
00160     return client.getMainCourseById(id);
00161   }
```

References client.

9.35.3.10 updateAppetizer()

```
void es.ull.esit.app.middleware.service.ProductService.updateAppetizer (
            Long id,
            String name,
            String priceStr)  [inline]
```

Updates an existing appetizer in the backend menu.

Parameters

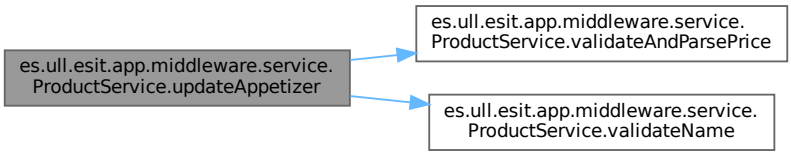
id	[Long] Identifier of the appetizer to update.
name	[String] New name for the appetizer.
priceStr	[String] New price entered as a string.

Definition at line 130 of file ProductService.java.

```
00130                                                                    {
00131     if (id == null) {
00132       throw new IllegalArgumentException("No appetizer selected!");
00133     }
00134
00135     int price = validateAndParsePrice(priceStr);
00136     validateName(name);
00137
00138     Appetizer updatedAppetizer = new Appetizer(id, name, price, null);
00139     client.updateAppetizer(id, updatedAppetizer);
00140   }
```

References client, validateAndParsePrice(), and validateName().

Here is the call graph for this function:



9.35.3.11 updateDrink()

```
void es.ull.esit.app.middleware.service.ProductService.updateDrink (
            Long id,
            String name,
            String priceStr)  [inline]
```

Updates an existing drink in the backend menu.

Parameters

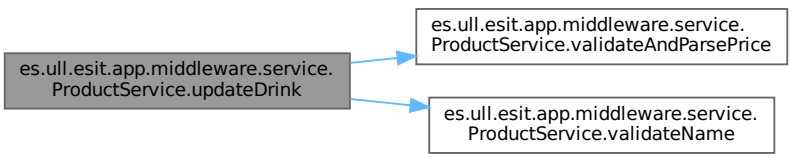
id	[Long] Identifier of the drink to update.
name	[String] New name for the drink.
priceStr	[String] New price entered as a string.

Definition at line 74 of file ProductService.java.

```
00074                                                                    {
00075     if (id == null) {
00076       throw new IllegalArgumentException("No drink selected!");
00077     }
00078
00079     int price = validateAndParsePrice(priceStr);
00080     validateName(name);
00081
00082     Drink updatedDrink = new Drink(id, name, price, null);
00083     client.updateDrink(id, updatedDrink);
00084   }
```

References client, validateAndParsePrice(), and validateName().

Here is the call graph for this function:



9.35.3.12 updateMainCourse()

```
void es.ull.esit.app.middleware.service.ProductService.updateMainCourse (
            Long id,
            String name,
            String priceStr)  [inline]
```

Updates an existing main course in the backend menu.

Parameters

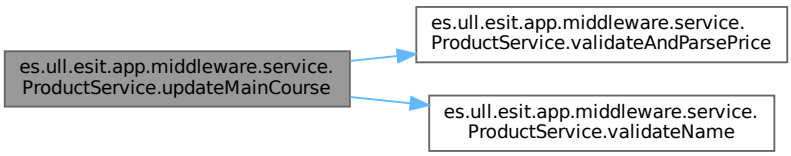
id	[Long] Identifier of the main course to update.
name	[String] New name for the main course.
priceStr	[String] New price entered as a string.

Definition at line 186 of file ProductService.java.

```
00186                                                                    {
00187     if (id == null) {
00188       throw new IllegalArgumentException("No main course selected!");
00189     }
00190
00191     int price = validateAndParsePrice(priceStr);
00192     validateName(name);
00193
00194     MainCourse updatedMainCourse = new MainCourse(id, name, price, null);
00195     client.updateMainCourse(id, updatedMainCourse);
00196   }
```

References client, validateAndParsePrice(), and validateName().

Here is the call graph for this function:



9.35.3.13 validateAndParsePrice()

```
int es.ull.esit.app.middleware.service.ProductService.validateAndParsePrice (
            String priceStr)  [inline], [private]
```

Validates and parses the price from string to integer.

Ensures the price is a non-negative integer value.

Parameters

priceStr	[String] Price entered as a string.

Returns

[int] Parsed price value.

Exceptions

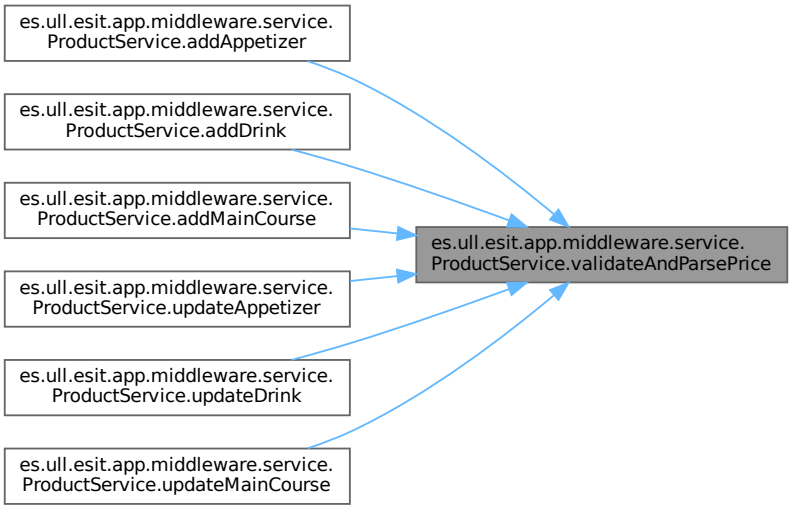
IllegalArgumentException	If the price is empty, negative or not a valid number.

Definition at line 221 of file ProductService.java.

```
00221                                                 {
00222     if (priceStr == null || priceStr.trim().isEmpty()) {
00223       throw new IllegalArgumentException("Price cannot be empty.");
00224     }
00225     try {
00226       int price = Integer.parseInt(priceStr.trim());
00227       if (price < 0) {
00228         throw new IllegalArgumentException("Price cannot be negative.");
00229       }
00230       return price;
00231     } catch (NumberFormatException e) {
00232       throw new IllegalArgumentException("Price must be a valid whole number.");
00233     }
00234   }
```

Referenced by addAppetizer(), addDrink(), addMainCourse(), updateAppetizer(), updateDrink(), and updateMainCourse().

Here is the caller graph for this function:



9.35.3.14 validateName()

```
void es.ull.esit.app.middleware.service.ProductService.validateName (
            String name)  [inline], [private]
```

Validates that the item name is not null or empty.

Parameters

name	[String] Name of the item to validate.

Exceptions

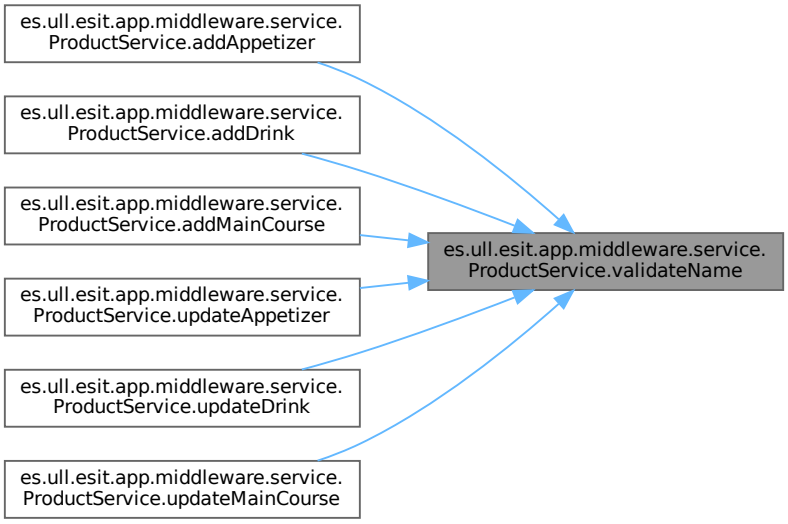
IllegalArgumentException	If the name is null or empty.

Definition at line 206 of file ProductService.java.

```
00206                                       {
00207     if (name == null || name.trim().isEmpty()) {
00208       throw new IllegalArgumentException("Item name cannot be empty.");
00209     }
00210   }
```

Referenced by addAppetizer(), addDrink(), addMainCourse(), updateAppetizer(), updateDrink(), and updateMainCourse().

Here is the caller graph for this function:



9.35.4 Member Data Documentation

9.35.4.1 client

```
final ApiClient es.ull.esit.app.middleware.service.ProductService.client  [private]
```

REST API client used to communicate with the backend menu endpoints.

Definition at line 19 of file ProductService.java.

Referenced by addAppetizer(), addDrink(), addMainCourse(), getAllAppetizers(), getAllDrinks(), getAllMainCourses(), getAppetizerById(), getDrinkById(), getMainCourseById(), ProductService(), updateAppetizer(), updateDrink(), and updateMainCourse().

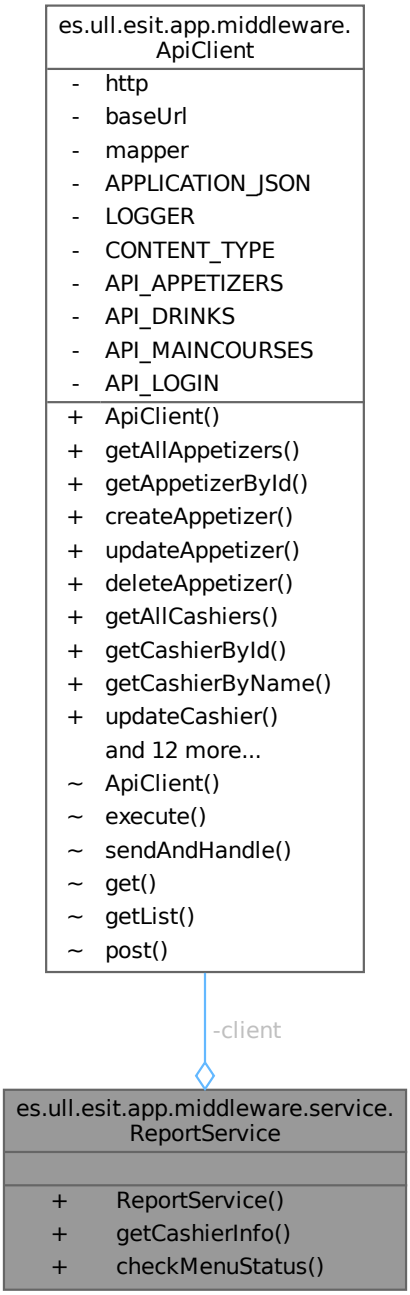
The documentation for this class was generated from the following file:

- ProductService.java

9.36 es.ull.esit.app.middleware.service.ReportService Class Reference

Service providing reporting and system status operations.

Collaboration diagram for es.ull.esit.app.middleware.service.ReportService:



Public Member Functions

- ReportService (ApiClient client)

Constructs a ReportService with the given API client.

- List< Cashier > getCashierInfo ()

  Loads information about all registered cashiers.

- String checkMenuStatus ()

  Checks backend connectivity and counts menu items.

Private Attributes

- final ApiClient client

  REST API client used to communicate with the backend.

9.36.1 Detailed Description

Service providing reporting and system status operations.

```
It offers methods to retrieve cashier information and to verify that
the menu endpoints of the backend are accessible.
```

Definition at line 16 of file ReportService.java.

9.36.2 Constructor & Destructor Documentation

9.36.2.1 ReportService()

```
es.ull.esit.app.middleware.service.ReportService.ReportService (
            ApiClient client)  [inline]
```

Constructs a ReportService with the given API client.

Parameters

client	[ApiClient] Client used to perform HTTP requests.

Definition at line 26 of file ReportService.java.

```
00026                                       {
00027     this.client = client;
00028   }
```

References client.

9.36.3 Member Function Documentation

9.36.3.1 checkMenuStatus()

```
String es.ull.esit.app.middleware.service.ReportService.checkMenuStatus ()  [inline]
```

Checks backend connectivity and counts menu items.

```
Performs GET requests to appetizers, drinks and main courses endpoints
and returns a textual summary of the available items.
```

Returns

[String] Human-readable status message about the menu system.

Definition at line 49 of file ReportService.java.

```
00049                                   {
00050     // Fetch lists to verify connectivity.
00051     List<Appetizer> appetizers = client.getAllAppetizers();
00052     List<Drink> drinks = client.getAllDrinks();
00053     List<MainCourse> mainCourses = client.getAllMainCourses();
00054
00055     return String.format(
00056         "Menu System Online: %d appetizers, %d drinks, %d main courses available.",
00057         appetizers.size(), drinks.size(), mainCourses.size());
00058   }
```

References client.

9.36.3.2 getCashierInfo()

```
List< Cashier > es.ull.esit.app.middleware.service.ReportService.getCashierInfo ()  [inline]
```

Loads information about all registered cashiers.

```
Uses the "/api/cashiers" endpoint to obtain the list.
```

Returns

[List<Cashier>] List of cashiers returned by the backend.

Definition at line 37 of file ReportService.java.

```
00037                                       {
00038     return client.getAllCashiers();
00039   }
```

References client.

9.36.4 Member Data Documentation

9.36.4.1 client

```
final ApiClient es.ull.esit.app.middleware.service.ReportService.client  [private]
```

REST API client used to communicate with the backend.

Definition at line 19 of file ReportService.java.

Referenced by checkMenuStatus(), getCashierInfo(), and ReportService().

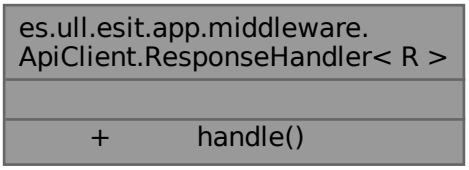
The documentation for this class was generated from the following file:

- ReportService.java

9.37 es.ull.esit.app.middleware.ApiClient.ResponseHandler< R > Interface Template Reference

Functional interface for processing HTTP responses and possibly throwing IOException from JSON parsing.

Collaboration diagram for es.ull.esit.app.middleware.ApiClient.ResponseHandler< R >:



Public Member Functions

- R handle (int status, String body) throws IOException

9.37.1 Detailed Description

Functional interface for processing HTTP responses and possibly throwing IOException from JSON parsing.

Definition at line 125 of file ApiClient.java.

9.37.2 Member Function Documentation

9.37.2.1 handle()

```
R es.ull.esit.app.middleware.ApiClient.ResponseHandler< R >.handle (
            int status,
            String body) throws IOException
```

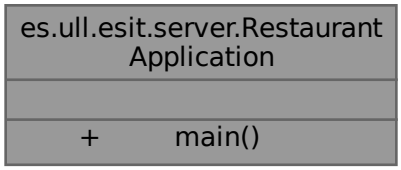
The documentation for this interface was generated from the following file:

- ApiClient.java

9.38 es.ull.esit.server.RestaurantApplication Class Reference

Main Spring Boot application class for the restaurant backend.

Collaboration diagram for es.ull.esit.server.RestaurantApplication:



Static Public Member Functions

- static void main (String[ ] args)

  Main entry point for the Spring Boot backend.

9.38.1 Detailed Description

Main Spring Boot application class for the restaurant backend.

```
Bootstraps the Spring Boot application.
Enables component scanning, auto-configuration and starts the embedded
web server that
exposes the REST controllers defined in the project.

The Swing frontend communicates with this backend via the HTTP
endpoints
provided by the controllers (Drinks, Appetizers, MainCourse, Menu,
etc.).
```

Definition at line 20 of file RestaurantApplication.java.

9.38.2 Member Function Documentation

9.38.2.1 main()

```
void es.ull.esit.server.RestaurantApplication.main (
            String[] args)  [inline], [static]
```

Main entry point for the Spring Boot backend.

```
Delegates to SpringApplication.run(), which starts the application
context, configures the environment and launches the embedded server.
```

Parameters

args	[String[]] Command-line arguments (not used).

Definition at line 30 of file RestaurantApplication.java.

```
00030                                       {
00031     SpringApplication.run(RestaurantApplication.class, args);
00032   }
```

The documentation for this class was generated from the following file:

- RestaurantApplication.java

9.39 es.ull.esit.server.config.SecurityConfig Class Reference

Spring Security configuration for the backend.

Collaboration diagram for es.ull.esit.server.config.SecurityConfig:

Public Member Functions

- PasswordEncoder passwordEncoder ()

  Creates a password encoder using BCrypt hashing algorithm.
- SecurityFilterChain filterChain (HttpSecurity http) throws Exception

  Configures the HTTP security settings:

9.39.1 Detailed Description

Spring Security configuration for the backend.

Defines:

- Password encoder used for hashing passwords.
- HTTP endpoint security policies: all endpoints are currently open (no authentication required).

The login logic is handled manually in the AuthController.

Definition at line 20 of file SecurityConfig.java.

9.39.2 Member Function Documentation

9.39.2.1 filterChain()

```
SecurityFilterChain es.ull.esit.server.config.SecurityConfig.filterChain (
            HttpSecurity http) throws Exception  [inline]
```

Configures the HTTP security settings:

- Disables CSRF protection.
- Permits unauthenticated access to specified endpoints (currently all).
- Enables HTTP Basic auth (username and password in headers).

Parameters

http	[HttpSecurity] The HttpSecurity builder to define security policies.

Returns

[SecurityFilterChain] The configured security filter chain.

Exceptions

Exception	If an error occurs during configuration.

Definition at line 45 of file SecurityConfig.java.

```
00045                                                                    {
00046     http
00047         .csrf().disable();
00048
00049     http
00050         .authorizeRequests()
00051         .anyRequest().permitAll();
00052
00053     http
00054         .httpBasic().disable()
00055         .formLogin().disable();
00056
00057     return http.build();
00058   }
```

9.39.2.2 passwordEncoder()

```
PasswordEncoder es.ull.esit.server.config.SecurityConfig.passwordEncoder ()  [inline]
```

Creates a password encoder using BCrypt hashing algorithm.

Returns

[PasswordEncoder] The BCrypt-based password encoder.

Definition at line 28 of file SecurityConfig.java.

```
00028                                       {
00029     return new BCryptPasswordEncoder();
00030   }
```

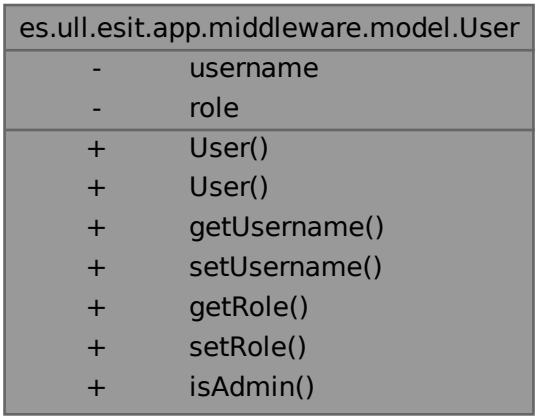
The documentation for this class was generated from the following file:

- SecurityConfig.java

9.40 es.ull.esit.app.middleware.model.User Class Reference

Client-side model representing an authenticated user.

Collaboration diagram for es.ull.esit.app.middleware.model.User:



Public Member Functions

- User ()

  Default constructor required for JSON deserialization.
- User (String username, String role)

  Constructs a user with the given username and role.
- String getUsername ()

  Gets the username.
- void setUsername (String username)

  Sets the username.
- String getRole ()

  Gets the user role.
- void setRole (String role)

  Sets the user role.
- boolean isAdmin ()

  Checks if the user has the ADMIN role.

Private Attributes

- String username

  Username of the authenticated user (JSON property "username").
- String role

  Role of the user (JSON property "role"), e.g., ADMIN or CASHIER.

9.40.1 Detailed Description

Client-side model representing an authenticated user.

```
It maps the JSON returned by the "/api/login" endpoint and contains
only the fields needed by the Swing application: username and role.
```

Definition at line 13 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

9.40.2 Constructor & Destructor Documentation

9.40.2.1 User() [1/2]

```
es.ull.esit.app.middleware.model.User.User () [inline]
```

Default constructor required for JSON deserialization.

Definition at line 26 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

```
00026                {
00027    }
```

References User().

Referenced by User().

Here is the call graph for this function:

Here is the caller graph for this function:

9.40.2.2 User() [2/2]

```
es.ull.esit.app.middleware.model.User.User (
            String username,
            String role)  [inline]
```

Constructs a user with the given username and role.

Parameters

username	[String] Username of the user.
role	[String] Role of the user.

Definition at line 35 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

```
00035                                           {
00036     this.username = username;
00037     this.role = role;
00038   }
```

References role, and username.

9.40.3 Member Function Documentation

9.40.3.1 getRole()

```
String es.ull.esit.app.middleware.model.User.getRole ()  [inline]
```

Gets the user role.

Returns

[String] Role of the user.

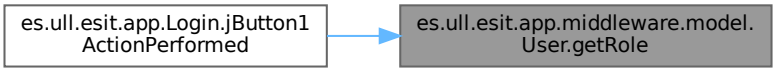
Definition at line 63 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

```
00063                           {
00064     return role;
00065   }
```

References role.

Referenced by es.ull.esit.app.Login.jButton1ActionPerformed().

Here is the caller graph for this function:



9.40.3.2 getUsername()

```
String es.ull.esit.app.middleware.model.User.getUsername ()  [inline]
```

Gets the username.

Returns

[String] Username of the user.

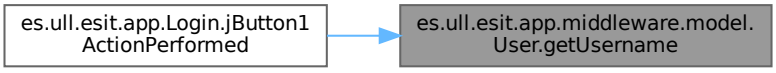
Definition at line 45 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

```
00045                               {
00046     return username;
00047   }
```

References username.

Referenced by es.ull.esit.app.Login.jButton1ActionPerformed().

Here is the caller graph for this function:



9.40.3.3 isAdmin()

```
boolean es.ull.esit.app.middleware.model.User.isAdmin ()  [inline]
```

Checks if the user has the ADMIN role.

Returns

[boolean] True if the user role is ADMIN (case-insensitive), false otherwise.

Definition at line 81 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

```
00081                           {
00082     return "ADMIN".equalsIgnoreCase(this.role);
00083   }
```

References role.

9.40.3.4 setRole()

```
void es.ull.esit.app.middleware.model.User.setRole (
             String role)  [inline]
```

Sets the user role.

Parameters

role	[String] Role of the user.

Definition at line 72 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

```
00072                                   {
00073     this.role = role;
00074   }
```

References role.

9.40.3.5 setUsername()

```
void es.ull.esit.app.middleware.model.User.setUsername (
             String username)  [inline]
```

Sets the username.

Parameters

username	[String] Username of the user.

Definition at line 54 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

```
00054                                       {
00055     this.username = username;
00056   }
```

References username.

9.40.4 Member Data Documentation

9.40.4.1 role

```
String es.ull.esit.app.middleware.model.User.role  [private]
```

Role of the user (JSON property "role"), e.g., ADMIN or CASHIER.

Definition at line 21 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

Referenced by getRole(), isAdmin(), setRole(), and User().

9.40.4.2 username

```
String es.ull.esit.app.middleware.model.User.username  [private]
```

Username of the authenticated user (JSON property "username").

Definition at line 17 of file src/main/java/es/ull/esit/app/middleware/model/User.java.

Referenced by getUsername(), setUsername(), and User().

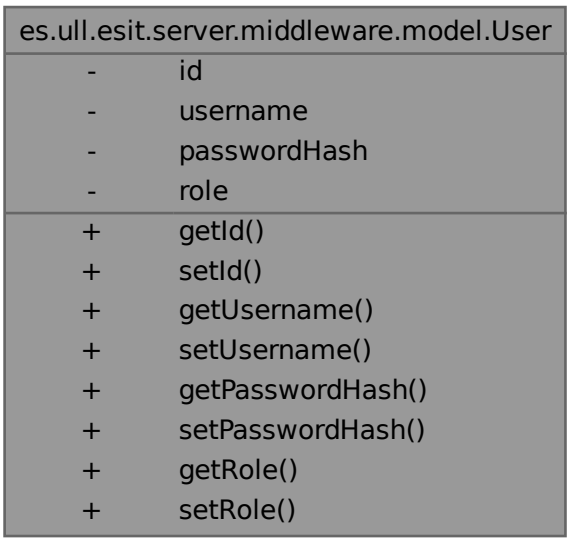
The documentation for this class was generated from the following file:

- src/main/java/es/ull/esit/app/middleware/model/User.java

9.41 es.ull.esit.server.middleware.model.User Class Reference

JPA entity that represents a user in the system.

Collaboration diagram for es.ull.esit.server.middleware.model.User:



Public Member Functions

- Long getId ()

  Gets the database identifier of the user.
- void setId (Long id)

  Sets the database identifier of the user.
- String getUsername ()

  Gets the user's username.
- void setUsername (String username)

  Sets the user's username.
- String getPasswordHash ()

  Gets the BCrypt hashed password of the user.
- void setPasswordHash (String passwordHash)

  Sets the BCrypt hashed password of the user.
- String getRole ()

  Gets the role of the user.
- void setRole (String role)

  Sets the role of the user.

Private Attributes

- Long id

  Unique identifier for the user (primary key).
- String username

  Unique username for the user.
- String passwordHash

  BCrypt hashed password for the user.
- String role

  Role of the user: ADMIN or CASHIER.

9.41.1 Detailed Description

JPA entity that represents a user in the system.

```
It is mapped to the "users" table in the database.
Stores user information such as username, password hash, and role.
```

Definition at line 22 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

9.41.2 Member Function Documentation

9.41.2.1 getId()

```
Long es.ull.esit.server.middleware.model.User.getId ()  [inline]
```

Gets the database identifier of the user.

Returns

[Long] The user's ID.

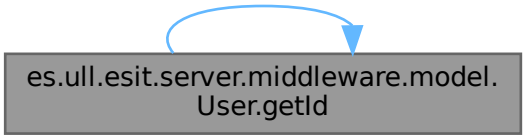
Definition at line 46 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00046                       {
00047     return id;
00048   }
```

References getId(), and id.

Referenced by getId().

Here is the call graph for this function:



Here is the caller graph for this function:

9.41.2.2 getPasswordHash()

```
String es.ull.esit.server.middleware.model.User.getPasswordHash ()  [inline]
```

Gets the BCrypt hashed password of the user.

Returns

[String] The user's BCrypt hashed password.

Definition at line 82 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00082                                {
00083     return passwordHash;
00084   }
```

References passwordHash.

9.41.2.3 getRole()

```
String es.ull.esit.server.middleware.model.User.getRole ()  [inline]
```

Gets the role of the user.

Returns

[String] The user's role.

Definition at line 100 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00100                          {
00101     return role;
00102   }
```

References role.

9.41.2.4 getUsername()

```
String es.ull.esit.server.middleware.model.User.getUsername ()  [inline]
```

Gets the user's username.

Returns

[String] The user's username.

Definition at line 64 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00064                            {
00065     return username;
00066   }
```

References username.

9.41.2.5 setId()


```
void es.ull.esit.server.middleware.model.User.setId (
            Long id)  [inline]
```

Parameters

id	[Long] The user's ID.

Definition at line 55 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00055                               {
00056     this.id = id;
00057   }
```

References id.

9.41.2.6 setPasswordHash()

```
void es.ull.esit.server.middleware.model.User.setPasswordHash (
             String passwordHash)  [inline]
```

Sets the BCrypt hashed password of the user.

Parameters

passwordHash	[String] The user's BCrypt hashed password.

Definition at line 91 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00091                                                {
00092     this.passwordHash = passwordHash;
00093   }
```

References passwordHash.

9.41.2.7 setRole()

```
void es.ull.esit.server.middleware.model.User.setRole (
             String role)  [inline]
```

Sets the role of the user.

Parameters

role	[String] The user's role.

Definition at line 109 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00109                                   {
00110     this.role = role;
00111   }
```

References role.

9.41.2.8 setUsername()

```
void es.ull.esit.server.middleware.model.User.setUsername (
             String username)  [inline]
```

Sets the user's username.

Parameters

username	[String] The user's username.

Definition at line 73 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

```
00073                                           {
00074     this.username = username;
00075   }
```

References username.

9.41.3 Member Data Documentation

9.41.3.1 id

`Long es.ull.esit.server.middleware.model.User.id  [private]`

Unique identifier for the user (primary key).

Definition at line 27 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

Referenced by getId(), and setId().

9.41.3.2 passwordHash

`String es.ull.esit.server.middleware.model.User.passwordHash  [private]`

BCrypt hashed password for the user.

Definition at line 36 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

Referenced by getPasswordHash(), and setPasswordHash().

9.41.3.3 role

`String es.ull.esit.server.middleware.model.User.role  [private]`

Role of the user: ADMIN or CASHIER.

Definition at line 39 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

Referenced by getRole(), and setRole().

9.41.3.4 username

`String es.ull.esit.server.middleware.model.User.username  [private]`

Unique username for the user.

Definition at line 31 of file server/src/main/java/es/ull/esit/server/middleware/model/User.java.

Referenced by getUsername(), and setUsername().

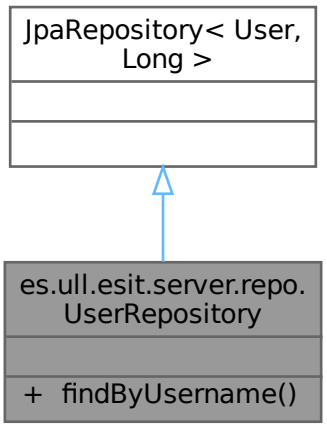
The documentation for this class was generated from the following file:

- server/src/main/java/es/ull/esit/server/middleware/model/User.java

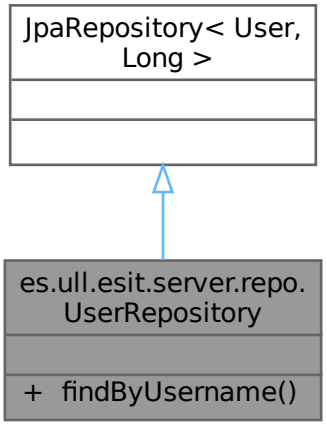
9.42 es.ull.esit.server.repo.UserRepository Interface Reference

Repository interface for accessing users in the database.

Inheritance diagram for es.ull.esit.server.repo.UserRepository:



Collaboration diagram for es.ull.esit.server.repo.UserRepository:



Public Member Functions

- Optional< User > findByUsername (String username)

  Finds a user by their unique username.

9.42.1 Detailed Description

Repository interface for accessing users in the database.

Extends JpaRepository to provide standard CRUD operations on the "users" table and defines an extra method to find a user by username.

Definition at line 13 of file UserRepository.java.

9.42.2 Member Function Documentation

9.42.2.1 findByUsername()

```
Optional< User > es.ull.esit.server.repo.UserRepository.findByUsername (
            String username)
```

Finds a user by their unique username.

Parameters

username	[String] The unique username of the user to find.

Returns

[Optional<User>] An Optional containing the User if found, or empty if not found.

The documentation for this interface was generated from the following file:

- UserRepository.java

Chapter 10

File Documentation

10.1 00-create-db.sql File Reference

10.2 00-create-db.sql

Go to the documentation of this file.

```
00001 -- 00-create-db.sql
00002 -- Crea la base de datos principal del proyecto.
00003
00004 CREATE DATABASE IF NOT EXISTS project3
00005   CHARACTER SET utf8mb4
00006   COLLATE utf8mb4_unicode_ci;
00007
00008 USE project3;
```

10.3 01-tables.sql File Reference

10.4 01-tables.sql

Go to the documentation of this file.

```
00001 -- 01-tables.sql
00002 -- Define todas las tablas de la base de datos.
00003
00004 USE project3;
00005
00006 -- =====================================
00007 -- USUARIOS DEL SISTEMA (para la API)
00008 -- =====================================
00009 CREATE TABLE IF NOT EXISTS users (
00010     id            BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
00011     username      VARCHAR(255) NOT NULL UNIQUE,
00012     password_hash VARCHAR(255) NOT NULL,
00013     role          VARCHAR(50)  NOT NULL
00014 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00015
00016 -- =====================================
00017 -- MANAGER
00018 -- =====================================
00019 CREATE TABLE IF NOT EXISTS manager (
00020     manager_id     BIGINT UNSIGNED PRIMARY KEY,
00021     manager_fname  VARCHAR(30),
00022     manager_mname  VARCHAR(30),
00023     manager_lname  VARCHAR(30),
00024     manager_number BIGINT,
00025     manager_salary INT
00026 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
00027
00028 -- =====================================
00029 -- CASHIER (vinculado a users.id)
00030 -- =====================================
00031 CREATE TABLE IF NOT EXISTS cashier (
00032     cashier_id     BIGINT UNSIGNED PRIMARY KEY,
00033     cashier_name   VARCHAR(30),
00034     cashier_salary INT,
00035     CONSTRAINT fk_cashier_user
00036       FOREIGN KEY (cashier_id) REFERENCES users(id)
00037 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00038
00039 -- =====================================
00040 -- CHEF
00041 -- =====================================
00042 CREATE TABLE IF NOT EXISTS chef (
00043     chef_id      BIGINT UNSIGNED PRIMARY KEY,
00044     chef_name    VARCHAR(30),
00045     chef_manager VARCHAR(30),
00046     chef_salary  INT,
00047     manager_id   BIGINT UNSIGNED,
00048     CONSTRAINT fk_chef_manager
00049       FOREIGN KEY (manager_id) REFERENCES manager (manager_id)
00050 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00051
00052 -- =====================================
00053 -- RECEIPT
00054 -- =====================================
00055 CREATE TABLE IF NOT EXISTS receipt (
00056     receipt_id    BIGINT UNSIGNED PRIMARY KEY,
00057     receipt_time  TIME,
00058     receipt_date  DATE,
00059     receipt_total INT
00060 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00061
00062 -- =====================================
00063 -- RESTAURANT MANAGEMENT
00064 -- =====================================
00065 CREATE TABLE IF NOT EXISTS restaurant_management (
00066     restaurant_id      BIGINT UNSIGNED PRIMARY KEY,
00067     restaurant_name    VARCHAR(50),
00068     restaurant_address VARCHAR(100),
00069     manager_id         BIGINT UNSIGNED,
00070     CONSTRAINT fk_restaurant_manager
00071       FOREIGN KEY (manager_id) REFERENCES manager (manager_id)
00072 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00073
00074 -- =====================================
00075 -- DEPENDENTS (MANAGER)
00076 -- =====================================
00077 CREATE TABLE IF NOT EXISTS dependent_manager (
00078     dependent_name     VARCHAR(30) PRIMARY KEY,
00079     dependent_relation VARCHAR(30),
00080     dependent_sex      ENUM('M','F'),
00081     manager_id         BIGINT UNSIGNED,
00082     CONSTRAINT fk_dep_manager
00083       FOREIGN KEY (manager_id) REFERENCES manager (manager_id)
00084 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00085
00086 -- =====================================
00087 -- DEPENDENTS (CHEF)
00088 -- =====================================
00089 CREATE TABLE IF NOT EXISTS dependent_chef (
00090     dependent_name     VARCHAR(30) PRIMARY KEY,
00091     dependent_relation VARCHAR(30),
00092     dependent_sex      ENUM('M','F'),
00093     chef_id            BIGINT UNSIGNED,
00094     CONSTRAINT fk_dep_chef
00095       FOREIGN KEY (chef_id) REFERENCES chef (chef_id)
00096 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00097
00098 -- =====================================
00099 -- DEPENDENTS (CASHIER)
00100 -- =====================================
00101 CREATE TABLE IF NOT EXISTS dependent_cashier (
00102     dependent_name     VARCHAR(30) PRIMARY KEY,
00103     dependent_relation VARCHAR(30),
00104     dependent_sex      ENUM('M','F'),
00105     cashier_id         BIGINT UNSIGNED,
00106     CONSTRAINT fk_dep_cashier
00107       FOREIGN KEY (cashier_id) REFERENCES cashier (cashier_id)
00108 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00109
00110 -- =====================================
00111 -- ITEM (productos combinados en un ticket)
00112 -- =====================================
00113 CREATE TABLE IF NOT EXISTS item (
```

```
00114     item_food         VARCHAR(30),
00115     food_price        INT,
00116     food_id           INT,
00117
00118     item_appetizers   VARCHAR(30),
00119     appetizers_price  INT,
00120     appetizers_id     INT,
00121
00122     item_drinks       VARCHAR(30),
00123     drinks_price      INT,
00124     drinks_id         INT,
00125
00126     receipt_id        BIGINT UNSIGNED,
00127
00128     PRIMARY KEY (food_id, appetizers_id, drinks_id),
00129     CONSTRAINT fk_item_receipt
00130       FOREIGN KEY (receipt_id) REFERENCES receipt (receipt_id)
00131 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00132
00133 CREATE INDEX item_index ON item (item_food);
00134
00135 -- =====================================
00136 -- CHEF_PREPARE_ITEM
00137 -- =====================================
00138 CREATE TABLE IF NOT EXISTS chef_prepare_item (
00139     chef_id   BIGINT UNSIGNED,
00140     food_id   INT,
00141     sweet_id  INT,
00142     drinks_id INT,
00143     CONSTRAINT fk_cpi_chef
00144       FOREIGN KEY (chef_id) REFERENCES chef (chef_id)
00145 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00146
00147 -- =====================================
00148 -- RECEIPT_TAKEN_BY_CASHIER
00149 -- =====================================
00150 CREATE TABLE IF NOT EXISTS receipt_taken_by_cashier (
00151     receipt_id BIGINT UNSIGNED,
00152     cashier_id BIGINT UNSIGNED,
00153     CONSTRAINT fk_rtc_receipt
00154       FOREIGN KEY (receipt_id) REFERENCES receipt (receipt_id),
00155     CONSTRAINT fk_rtc_cashier
00156       FOREIGN KEY (cashier_id) REFERENCES cashier (cashier_id)
00157 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00158
00159 -- =====================================
00160 -- RESTAURANT_ADDRESS
00161 -- =====================================
00162 CREATE TABLE IF NOT EXISTS restaurant_address (
00163     restaurant_address VARCHAR(100) PRIMARY KEY,
00164     restaurant_id      BIGINT UNSIGNED,
00165     CONSTRAINT fk_restaurant_addr
00166       FOREIGN KEY (restaurant_id) REFERENCES restaurant_management (restaurant_id)
00167 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00168
00169 -- =====================================
00170 -- TABLAS DE PRODUCTOS PARA LA API
00171 -- =====================================
00172 CREATE TABLE IF NOT EXISTS appetizers (
00173   id    INT NOT NULL AUTO_INCREMENT,
00174   name  VARCHAR(250) NOT NULL,
00175   price INT NOT NULL,
00176   PRIMARY KEY (id)
00177 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00178
00179 CREATE TABLE IF NOT EXISTS drinks (
00180   id    INT NOT NULL AUTO_INCREMENT,
00181   name  VARCHAR(250) NOT NULL,
00182   price INT NOT NULL,
00183   PRIMARY KEY (id)
00184 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00185
00186 CREATE TABLE IF NOT EXISTS maincourse (
00187   id    INT NOT NULL AUTO_INCREMENT,
00188   name  VARCHAR(250) NOT NULL,
00189   price INT NOT NULL,
00190   PRIMARY KEY (id)
00191 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
00192
00193 -- =====================================
00194 -- TABLA DE PRUEBAS: receipt_copy
00195 -- =====================================
00196 CREATE TABLE IF NOT EXISTS receipt_copy (
00197     receipt_id    BIGINT UNSIGNED PRIMARY KEY,
00198     receipt_time  TIME,
00199     receipt_date  DATE,
00200     receipt_total INT
```

```
00201 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

10.5 02-procedures.sql File Reference

10.6 02-procedures.sql

Go to the documentation of this file.

```
00001 -- 02-procedures.sql
00002 -- Define procedimientos almacenados de la base de datos.
00003
00004 USE project3;
00005
00006 DROP PROCEDURE IF EXISTS chefname;
00007
00008 DELIMITER $$
00009
00010 CREATE PROCEDURE chefname (IN chef_name_param VARCHAR(10))
00011 BEGIN
00012     SELECT *
00013     FROM chef
00014     WHERE chef_name = chef_name_param;
00015 END $$
00016
00017 DELIMITER ;
00018
```

10.7 03-triggers.sql File Reference

10.8 03-triggers.sql

Go to the documentation of this file.

```
00001 -- 03-triggers.sql
00002 -- Añade triggers a la base de datos.
00003
00004 USE project3;
00005
00006 -- =====================================
00007 -- Trigger: nombres de manager en mayúsculas
00008 -- =====================================
00009 DROP TRIGGER IF EXISTS trg_uppercase_manager_fname;
00010
00011 DELIMITER $$
00012
00013 CREATE TRIGGER trg_uppercase_manager_fname
00014 BEFORE INSERT ON manager
00015 FOR EACH ROW
00016 BEGIN
00017   SET NEW.manager_fname = UPPER(NEW.manager_fname);
00018 END $$
00019
00020 DELIMITER ;
00021
00022 -- =====================================
00023 -- Trigger: crear cashier automáticamente al insertar un usuario CASHIER
00024 -- Asigna salario base = 1500 €
00025 -- =====================================
00026 DROP TRIGGER IF EXISTS trg_create_cashier_from_user;
00027
00028 DELIMITER $$
00029
00030 CREATE TRIGGER trg_create_cashier_from_user
00031 AFTER INSERT ON users
00032 FOR EACH ROW
00033 BEGIN
00034   IF NEW.role = 'CASHIER' THEN
00035     INSERT INTO cashier (cashier_id, cashier_name, cashier_salary)
00036     VALUES (NEW.id, NEW.username, 1500);
00037   END IF;
00038 END $$
00039
00040 DELIMITER ;
00041
```

10.9 04-privileges.sql File Reference

10.10 04-privileges.sql

Go to the documentation of this file.

```
00001 -- 04-privileges.sql
00002 -- Asigna permisos al usuario usado por Spring Boot.
00003
00004 USE project3;
00005
00006 GRANT ALL PRIVILEGES ON project3.* TO 'restaurant'@'%';
00007 FLUSH PRIVILEGES;
```

10.11 05-data.sql File Reference

10.12 05-data.sql

Go to the documentation of this file.

```
00001 -- 05-data.sql
00002 -- Inserción de datos realistas para el restaurante "Black Plate"
00003
00004 USE project3;
00005
00006 -- ========================================================================
00007 -- MANAGER
00008 -- ========================================================================
00009 INSERT INTO manager (manager_id, manager_fname, manager_mname, manager_lname, manager_number,
      manager_salary)
00010 VALUES
00011 (1001, 'marta', 'elena', 'rodriguez', 659123456, 3200);
00012
00013 -- ========================================================================
00014 -- RESTAURANT
00015 -- ========================================================================
00016 INSERT INTO restaurant_management (restaurant_id, restaurant_name, restaurant_address, manager_id)
00017 VALUES
00018 (2001, 'black plate', 'Calle Luna 45, Madrid', 1001);
00019
00020 -- ========================================================================
00021 -- CHEFS DE BLACK PLATE
00022 -- ========================================================================
00023 INSERT INTO chef (chef_id, chef_name, chef_manager, chef_salary, manager_id)
00024 VALUES
00025 (3001, 'Luis Gómez', 'Marta', 1800, 1001),
00026 (3002, 'Carmen Ruiz', 'Marta', 1750, 1001),
00027 (3003, 'Javier Soto', 'Marta', 1700, 1001);
00028
00029 -- ========================================================================
00030 -- RECEIPTS (Tickets reales)
00031 -- ========================================================================
00032 INSERT INTO receipt (receipt_id, receipt_time, receipt_date, receipt_total)
00033 VALUES
00034 (1, '13:15:20', '2024-10-01', 42),
00035 (2, '14:22:10', '2024-10-01', 27),
00036 (3, '20:45:50', '2024-10-02', 58),
00037 (4, '21:10:35', '2024-10-03', 73),
00038 (5, '12:30:10', '2024-10-04', 19);
00039
00040 -- ========================================================================
00041 -- DEPENDENTS DEL MANAGER
00042 -- ========================================================================
00043 INSERT INTO dependent_manager (dependent_name, dependent_relation, dependent_sex, manager_id)
00044 VALUES
00045 ('lucia', 'daughter', 'F', 1001),
00046 ('pablo', 'son',      'M', 1001);
00047
00048 -- ========================================================================
00049 -- DEPENDENTS DE CHEF
00050 -- ========================================================================
00051 INSERT INTO dependent_chef (dependent_name, dependent_relation, dependent_sex, chef_id)
00052 VALUES
00053 ('marcos', 'son',  'M', 3001),
```

```
00054 ('irene',  'wife', 'F', 3002);
00055
00056 -- ========================================================================
00057 -- USUARIOS DEL SISTEMA
00058 -- El trigger creará automáticamente los CASHIERS en la tabla cashier
00059 -- ========================================================================
00060
00061 -- ADMIN (NO genera cashier)
00062 -- password: admin
00063 INSERT INTO users (id, username, password_hash, role)
00064 VALUES (1, 'admin', '$2b$10$TAxhf.ziwuv7ynXlSQYjMeftWo47ft8M4JfdXjLoBPdERwuo7zD06', 'ADMIN');
00065
00066 -- CAJEROS (generan automáticamente filas en cashier gracias al trigger)
00067 -- password: cashier123 (hash reutilizado)
00068 INSERT INTO users (id, username, password_hash, role)
00069 VALUES
00070 (2, 'laura.mendez', '$2b$10$HLeNcr00JL6qpGJUd.klqud4PfSRci2qvuKr7fnklGjtMOO6OhwLC', 'CASHIER'),
00071 (3, 'diego.santos', '$2b$10$HLeNcr00JL6qpGJUd.klqud4PfSRci2qvuKr7fnklGjtMOO6OhwLC', 'CASHIER');
00072
00073 -- En este punto, gracias al trigger:
00074 --   cashier tendrá:
00075 --   (2, 'laura.mendez', 1500) y (3, 'diego.santos', 1500)
00076
00077 -- ========================================================================
00078 -- DEPENDENTS DE LOS CASHIERS
00079 -- ========================================================================
00080 INSERT INTO dependent_cashier (dependent_name, dependent_relation, dependent_sex, cashier_id)
00081 VALUES
00082 ('emma', 'daughter', 'F', 2),
00083 ('alex', 'son',      'M', 3);
00084
00085 -- ========================================================================
00086 -- ITEMS DEL TICKET (combinaciones reales del menú Black Plate)
00087 -- ========================================================================
00088 INSERT INTO item (item_food, food_price, food_id,
00089                   item_appetizers, appetizers_price, appetizers_id,
00090                   item_drinks, drinks_price, drinks_id,
00091                   receipt_id)
00092 VALUES
00093 ('Grilled Salmon',      18, 1, 'Truffle Fries',      6, 1, 'Coca Cola',       2, 1, 1),
00094 ('Black Angus Burger',  16, 2, 'Garlic Bread',       4, 2, 'Sparkling Water', 2, 2, 2),
00095 ('Chicken Teriyaki',    14, 3, 'Hummus Bowl',        5, 3, 'Iced Tea',        3, 3, 3),
00096 ('Vegan Buddha Bowl',   13, 4, 'Sweet Potato Chips', 4, 4, 'Lemonade',        3, 4, 4),
00097 ('Pasta Alfredo',       11, 5, 'Bruschetta',         4, 5, 'Coca Cola',       2, 1, 5);
00098
00099 -- ========================================================================
00100 -- CHEF PREPARA PLATOS
00101 -- ========================================================================
00102 INSERT INTO chef_prepare_item (chef_id, food_id, sweet_id, drinks_id)
00103 VALUES
00104 (3001, 1, 1, 1),
00105 (3002, 2, 2, 2),
00106 (3003, 3, 3, 3);
00107
00108 -- ========================================================================
00109 -- RECEIPTS TOMADOS POR CASHIERS (usa IDs 2 y 3)
00110 -- ========================================================================
00111 INSERT INTO receipt_taken_by_cashier (receipt_id, cashier_id)
00112 VALUES
00113 (1, 2),
00114 (2, 2),
00115 (3, 3),
00116 (4, 3),
00117 (5, 2);
00118
00119 -- ========================================================================
00120 -- Tabla receipt_copy (pruebas)
00121 -- ========================================================================
00122 INSERT INTO receipt_copy (receipt_id, receipt_time, receipt_date, receipt_total)
00123 VALUES
00124 (1, '12:45:51', '2024-10-01', 42);
00125
00126 DELETE FROM receipt_copy WHERE receipt_id = 1;
00127
00128 -- ========================================================================
00129 -- Productos para la API (Black Plate)
00130 -- ========================================================================
00131 INSERT INTO appetizers (name, price) VALUES
00132 ('Truffle Fries',      6),
00133 ('Garlic Bread',       4),
00134 ('Hummus Bowl',        5),
00135 ('Sweet Potato Chips', 4),
00136 ('Bruschetta',         4);
00137
00138 INSERT INTO drinks (name, price) VALUES
00139 ('Coca Cola',       2),
00140 ('Sparkling Water', 2),
```

```
00141 ('Iced Tea',        3),
00142 ('Lemonade',        3),
00143 ('Red Fruit Juice', 4);
00144
00145 INSERT INTO maincourse (name, price) VALUES
00146 ('Grilled Salmon',        18),
00147 ('Black Angus Burger',    16),
00148 ('Chicken Teriyaki',      14),
00149 ('Vegan Buddha Bowl',     13),
00150 ('Pasta Alfredo',         11);
```

10.13 init.sql File Reference

10.14 init.sql

Go to the documentation of this file.

```
00001 /**
00002  * init.sql
00003  * Script de inicialización de la base de datos project3 para el contenedor Docker MySQL.
00004  */
00005
00006 -- Desactivar restricciones de claves foráneas.
00007 SET FOREIGN_KEY_CHECKS = 0;
00008
00009 -- Crear BD si no existe.
00010 CREATE DATABASE IF NOT EXISTS project3;
00011 USE project3;
00012
00013 ----- CREACIÓN DE TABLAS -----
00014 -- Tabla 1. RestaurantManagement.
00015 CREATE TABLE IF NOT EXISTS RestaurantManagement (
00016     Restaurant_id      NUMERIC(10) PRIMARY KEY,
00017     Restaurant_name    VARCHAR(30),
00018     Restaurant_address VARCHAR(30),
00019     Manager_id         NUMERIC(10)
00020         REFERENCES Manager (Manager_id)
00021 );
00022
00023 -- Tabla 2. Manager.
00024 CREATE TABLE IF NOT EXISTS Manager (
00025     Manager_id     NUMERIC(10) PRIMARY KEY,
00026     Manager_Fname  VARCHAR(30),
00027     Manager_Mname  VARCHAR(30),
00028     Manager_Lname  VARCHAR(30),
00029     Manager_number NUMERIC(10),
00030     Manager_salary NUMERIC(5)
00031 );
00032
00033 -- Tabla 3. Item.
00034 CREATE TABLE IF NOT EXISTS Item (
00035     Item_food         VARCHAR(30),
00036     food_price        NUMERIC(4),
00037     food_id           NUMERIC(5),
00038
00039     Item_appetizers   VARCHAR(30),
00040     appetizers_price  NUMERIC(4),
00041     appetizers_id     NUMERIC(5),
00042
00043     Item_drinks       VARCHAR(30),
00044     drinks_price      NUMERIC(4),
00045     drinks_id         NUMERIC(5),
00046
00047     Receipt_id NUMERIC(10) REFERENCES Receipt (Receipt_id),
00048
00049     PRIMARY KEY (food_id, appetizers_id, drinks_id)
00050 );
00051
00052 -- Tabla 4. Chef.
00053 CREATE TABLE IF NOT EXISTS Chef (
00054     Chef_id      NUMERIC(10) PRIMARY KEY,
00055     Chef_name    VARCHAR(30),
00056     Chef_manager VARCHAR(30),
00057     Chef_salary  NUMERIC(5),
00058     Manager_id   NUMERIC(10) REFERENCES Manager (Manager_id)
00059 );
00060
00061 -- Tabla 5. Cashier.
00062 CREATE TABLE IF NOT EXISTS Cashier (
```

```
00063     Cashier_id     NUMERIC(10) PRIMARY KEY,
00064     Cashier_name   VARCHAR(30),
00065     Cashier_salary NUMERIC(5)
00066 );
00067
00068 -- Tabla 6. Receipt.
00069 CREATE TABLE IF NOT EXISTS Receipt (
00070     Receipt_id    NUMERIC(10) PRIMARY KEY,
00071     Receipt_time  TIME,
00072     Receipt_date  DATE,
00073     Receipt_total NUMERIC(10)
00074 );
00075
00076 -- Tabla 7. Dependent_Manager.
00077 CREATE TABLE IF NOT EXISTS Dependent_Manager (
00078     Dependent_name     VARCHAR(30) PRIMARY KEY,
00079     Dependent_relation VARCHAR(30),
00080     Dependent_sex      ENUM('M','F'),
00081     Manager_id         NUMERIC(10) REFERENCES Manager (Manager_id)
00082 );
00083
00084 -- Tabla 8. Dependent_chef.
00085 CREATE TABLE IF NOT EXISTS Dependent_chef (
00086     Dependent_name     VARCHAR(30) PRIMARY KEY,
00087     Dependent_relation VARCHAR(30),
00088     Dependent_sex      ENUM('M','F'),
00089     Chef_id            NUMERIC(10) REFERENCES Chef (Chef_id)
00090 );
00091
00092 -- Tabla 9. Dependent_Cashier.
00093 CREATE TABLE IF NOT EXISTS Dependent_Cashier (
00094     Dependent_name     VARCHAR(30) PRIMARY KEY,
00095     Dependent_relation VARCHAR(30),
00096     Dependent_sex      ENUM('M','F'),
00097     Cashier_id         NUMERIC(10) REFERENCES Cashier (Cashier_id)
00098 );
00099
00100 -- Tabla 10. Restaurant_address.
00101 CREATE TABLE IF NOT EXISTS Restaurant_address (
00102     Restaurant_address VARCHAR(30) PRIMARY KEY,
00103     Restaurant_id      NUMERIC(10) REFERENCES RestaurantManagement (Restaurant_id)
00104 );
00105
00106 -- Tabla 11. Chef_Prepare_item.
00107 CREATE TABLE IF NOT EXISTS Chef_Prepare_item (
00108     Chef_id   NUMERIC(10) REFERENCES Chef (Chef_id),
00109     food_id   NUMERIC(5) REFERENCES Item (food_id),
00110     sweet_id  NUMERIC(5),
00111     drinks_id NUMERIC(5)
00112 );
00113
00114 -- Tabla 12. Receipt_takenBy_Cashier.
00115 CREATE TABLE IF NOT EXISTS Receipt_takenBy_Cashier (
00116     Receipt_id NUMERIC(10) REFERENCES Receipt (Receipt_id),
00117     Cashier_id NUMERIC(10) REFERENCES Cashier (Cashier_id)
00118 );
00119
00120 ----- INSERCIÓN DE DATOS -----
00121 -- Restaurant --
00122 INSERT INTO RestaurantManagement VALUES (56471,'project_resturant','khobar',3214);
00123
00124 -- Manager --
00125 INSERT INTO Manager VALUES
00126 (3214,'ahmed','khalid','alfahad',053276488,7000);
00127
00128 -- Chefs --
00129 INSERT INTO Chef VALUES
00130 (2561,'abduallah','ahmed',6000,3214),
00131 (2562,'saleh','ahmed',6000,3214),
00132 (2563,'mohammad','ahmed',6000,3214),
00133 (2564,'khalid','ahmed',6000,3214);
00134
00135 -- Cashiers --
00136 INSERT INTO Cashier VALUES
00137 (4231,'abdualmajeed',7000),
00138 (4232,'abdualrahman',7000);
00139
00140 -- Receipts --
00141 INSERT INTO Receipt VALUES
00142 (1,'12:45:56','2022-01-23',300),
00143 (2,'12:44:32','2022-01-23',200),
00144 (3,'01:30:50','2022-01-24',300),
00145 (4,'03:30:55','2022-01-24',360),
00146 (5,'03:00:50','2022-01-25',400),
00147 (6,'02:00:00','2022-01-25',200),
00148 (7,'03:00:00','2022-01-26',150);
00149
```

```
00150 -- Dependents (Manager) --
00151 INSERT INTO Dependent_Manager VALUES
00152 ('maram','wife','F',3214),
00153 ('marwa','child','F',3214);
00154
00155 -- Dependents (Chef) --
00156 INSERT INTO Dependent_chef VALUES
00157 ('saad','son','M',2562),
00158 ('sara','wife','F',2563),
00159 ('norah','wife','F',2561);
00160
00161 -- Dependents (Cashier) --
00162 INSERT INTO Dependent_Cashier VALUES
00163 ('sara','daughter','F',4231),
00164 ('lama','daughter','F',4232);
00165
00166 -- Items --
00167 INSERT INTO Item VALUES
00168 ('buratta pizza',56,1,'Dynamite shrimp',39,1,'cola',5,1,1),
00169 ('pink pasta',37,2,'mac&cheese balls',45,2,'7up',5,2,2),
00170 ('spaghetti',40,3,'tiramisu',42,3,'orang juice',15,3,3),
00171 ('rosemary salmon',87,4,'molten chocolate',19,4,'mojito',25,4,4);
00172
00173 -- Chef Prepare Item --
00174 INSERT INTO Chef_Prepare_item VALUES
00175 (2561,1,1,1),
00176 (2562,2,2,2),
00177 (2563,3,3,3),
00178 (2564,4,4,4);
00179
00180 -- Receipt taken by cashier --
00181 INSERT INTO Receipt_takenBy_Cashier VALUES
00182 (1,1),(2,1),(3,1),(4,1),(5,2),(6,2),(7,2);
00183
00184 -- CONSULTAS, PRUEBAS Y PROCEDIMIENTOS --
00185 -- SELECTs de prueba --
00186 SELECT DISTINCT Receipt_total FROM Receipt;
00187
00188 SELECT AVG(Receipt_total) AS avg_total FROM Receipt;
00189 SELECT MAX(Receipt_total) AS max_total FROM Receipt;
00190 SELECT MIN(Receipt_total) AS min_total FROM Receipt;
00191
00192 SELECT * FROM Chef WHERE NOT Chef_name='khalid';
00193 SELECT * FROM Chef WHERE Chef_name LIKE 'k%';
00194
00195 SELECT * FROM Item ORDER BY Item_food DESC;
00196 SELECT * FROM Item ORDER BY Item_food ASC;
00197
00198 SELECT COUNT(Chef_id), Chef_name FROM Chef GROUP BY Chef_name HAVING COUNT(Chef_id) > 2561;
00199
00200 SELECT * FROM Receipt WHERE Receipt_date IN ('2022-01-24','2022-01-25');
00201 SELECT * FROM Receipt WHERE Receipt_total BETWEEN 300 AND 400;
00202
00203 -- CASE example --
00204 INSERT INTO Receipt VALUES (10,'13:46:51','2022-02-23',650);
00205
00206 SELECT
00207     Receipt_id,
00208     Receipt_time,
00209     Receipt_date,
00210     Receipt_total,
00211     CASE
00212         WHEN Receipt_total BETWEEN 600 AND 700 THEN 'salary increasing'
00213         ELSE 'normal'
00214     END AS salary
00215 FROM Receipt;
00216
00217 -- Receipt copy table --
00218 CREATE TABLE IF NOT EXISTS Receipt_copy (
00219     Receipt_id    NUMERIC(10) PRIMARY KEY,
00220     Receipt_time  TIME,
00221     Receipt_date  DATE,
00222     Receipt_total NUMERIC(10)
00223 );
00224
00225 INSERT INTO Receipt_copy VALUES (1,'12:45:51','2022-01-23',300);
00226 DELETE FROM Receipt_copy WHERE Receipt_id = 1;
00227
00228 -- Index --
00229 CREATE INDEX Itemindex ON Item (Item_food);
00230
00231 -- PROCEDURE: Chefname --
00232 DELIMITER $$
00233 CREATE PROCEDURE Chefname (IN Cheffname VARCHAR(10))
00234 BEGIN
00235     SELECT * FROM Chef WHERE Chef_name = Cheffname;
00236 END $$
```

```
00237 DELIMITER ;
00238
00239 CALL Chefname('mohammad');
00240 CALL Chefname('abduallah');
00241
00242 -- TRIGGER: uppercase names in Manager --
00243 CREATE TRIGGER uppercase BEFORE INSERT ON Manager
00244 FOR EACH ROW
00245 SET NEW.Manager_Fname = UPPER(NEW.Manager_Fname);
00246
00247 INSERT INTO Manager VALUES (3217,'Yazeed','fahad','alahmad',053276488,7000);
00248
00249 SELECT * FROM Manager;
00250
00251 -- CREACIÓN E INSERCIÓN DE DATOS EN TABLAS ADICIONALES --
00252 -- Tabla Appetizers --
00253 CREATE TABLE IF NOT EXISTS appetizers (
00254   id    INT NOT NULL AUTO_INCREMENT,
00255   name  VARCHAR(250) NOT NULL,
00256   price INT NOT NULL,
00257   PRIMARY KEY (id)
00258 );
00259
00260 INSERT INTO appetizers (id, name, price) VALUES
00261 (1,'Truffel Fries',23),
00262 (2,'Molten Chocolate',12),
00263 (3,'Mac&Cheese Balls',12),
00264 (4,'Dynamite Shrimp',32),
00265 (5,'Kheera',10);
00266
00267 -- Tabla Drinks --
00268 CREATE TABLE IF NOT EXISTS drinks (
00269   id    INT NOT NULL AUTO_INCREMENT,
00270   name  VARCHAR(250) NOT NULL,
00271   price INT NOT NULL,
00272   PRIMARY KEY (id)
00273 );
00274
00275 INSERT INTO drinks (id, name, price) VALUES
00276 (1,'cola',6),
00277 (2,'7up',6),
00278 (3,'orange juice',10),
00279 (4,'mojito',14),
00280 (5,'Red Bull',8);
00281
00282 -- Tabla MainCourse --
00283 CREATE TABLE IF NOT EXISTS maincourse (
00284   id    INT NOT NULL AUTO_INCREMENT,
00285   name  VARCHAR(250) NOT NULL,
00286   price INT NOT NULL,
00287   PRIMARY KEY (id)
00288 );
00289
00290 INSERT INTO maincourse (id, name, price) VALUES
00291 (1,'Buratta Pizza',54),
00292 (2,'Pink Pasta',12),
00293 (3,'Rosemary Salmon',30),
00294 (4,'Spaghetti',8),
00295 (5,'Crown Pizza',50);
00296
00297 COMMIT;
00298
00299 -- CONCESIÓN DE PRIVILEGIOS AL USUARIO RESTAURANT --
00300 GRANT ALL PRIVILEGES ON project3.* TO 'restaurant'@'%';
00301 FLUSH PRIVILEGES;
00302
00303 -- Reactivar restricciones de claves foráneas. --
00304 SET FOREIGN_KEY_CHECKS = 1;
```

10.15 README.md File Reference

10.16 SecurityConfig.java File Reference

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.Http↩
Security;
```

```
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
```

Include dependency graph for SecurityConfig.java:

Classes

- class es.ull.esit.server.config.SecurityConfig

  Spring Security configuration for the backend.

Packages

- package es.ull.esit.server.config

10.17 SecurityConfig.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.config;
00002
00003 import org.springframework.context.annotation.Bean;
00004 import org.springframework.context.annotation.Configuration;
00005 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
00006 import org.springframework.security.web.SecurityFilterChain;
00007 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
00008 import org.springframework.security.crypto.password.PasswordEncoder;
00009
00019 @Configuration
00020 public class SecurityConfig {
00021
00027   @Bean
00028   public PasswordEncoder passwordEncoder() {
00029     return new BCryptPasswordEncoder();
00030   }
00031
00044   @Bean
00045   public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
00046     http
00047         .csrf().disable();
00048
00049     http
00050         .authorizeRequests()
00051         .anyRequest().permitAll();
00052
00053     http
00054         .httpBasic().disable()
00055         .formLogin().disable();
00056
00057     return http.build();
00058   }
00059 }
```

10.18 AppetizerController.java File Reference

```
import es.ull.esit.server.middleware.model.Appetizer;
import es.ull.esit.server.repo.AppetizerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;
```

Include dependency graph for AppetizerController.java:

Classes

- class es.ull.esit.server.controller.AppetizerController

  REST controller for managing appetizers.

Packages

- package es.ull.esit.server.controller

10.19 AppetizerController.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.controller;
00002
00003 import es.ull.esit.server.middleware.model.Appetizer;
00004 import es.ull.esit.server.repo.AppetizerRepository;
00005 import org.springframework.beans.factory.annotation.Autowired;
00006 import org.springframework.http.*;
00007 import org.springframework.web.bind.annotation.*;
00008
00009 import java.util.List;
00010 import java.util.Optional;
00011
00019 @RestController
00020 @RequestMapping("/api/appetizers")
00021 public class AppetizerController {
00022
00024   @Autowired
00025   private AppetizerRepository appetizerRepository;
00026
00034   @GetMapping
00035   public ResponseEntity<List<Appetizer>> getAllAppetizers() {
00036     List<Appetizer> appetizers = appetizerRepository.findAll();
00037     return ResponseEntity.ok(appetizers);
00038   }
00039
00048   @GetMapping("/{id}")
00049   public ResponseEntity<Appetizer> getAppetizerById(@PathVariable Long id) {
00050     Optional<Appetizer> appetizer = appetizerRepository.findById(id);
00051     return appetizer
00052         .map(ResponseEntity::ok)
00053         .orElseGet(() -> ResponseEntity.notFound().build());
00054   }
00055
00064   @PostMapping
00065   public ResponseEntity<Appetizer> createAppetizer(@RequestBody Appetizer appetizer) {
00066     Appetizer saved = appetizerRepository.save(appetizer);
```

```
00067     return ResponseEntity.status(HttpStatus.CREATED).body(saved);
00068   }
00069
00079   @PutMapping("/{id}")
00080   public ResponseEntity<Appetizer> updateAppetizer(@PathVariable Long id,
00081                                                    @RequestBody Appetizer appetizer) {
00082     if (!appetizerRepository.existsById(id)) {
00083       return ResponseEntity.notFound().build();
00084     }
00085     appetizer.setAppetizersId(id);
00086     Appetizer updated = appetizerRepository.save(appetizer);
00087     return ResponseEntity.ok(updated);
00088   }
00089
00098   @DeleteMapping("/{id}")
00099   public ResponseEntity<Void> deleteAppetizer(@PathVariable Long id) {
00100     if (!appetizerRepository.existsById(id)) {
00101       return ResponseEntity.notFound().build();
00102     }
00103     appetizerRepository.deleteById(id);
00104     return ResponseEntity.noContent().build();
00105   }
00106 }
```

10.20 AuthController.java File Reference

```
import es.ull.esit.server.middleware.model.User;
import es.ull.esit.server.repo.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.*;
import java.util.Optional;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

Include dependency graph for AuthController.java:

Classes

- class es.ull.esit.server.controller.AuthController

  Rest controller for handling authentication-related requests.
- class es.ull.esit.server.controller.AuthController.LoginRequest

  Simple DTO (Data Transfer Object) for login requests payload.

Packages

- package es.ull.esit.server.controller

10.21 AuthController.java

Go to the documentation of this file.

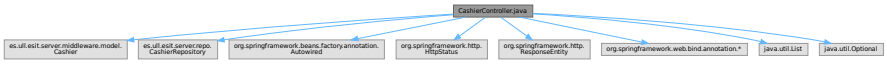
```
00001 package es.ull.esit.server.controller;
00002
00003 import es.ull.esit.server.middleware.model.User;
00004 import es.ull.esit.server.repo.UserRepository;
00005
00006 import org.springframework.beans.factory.annotation.Autowired;
00007 import org.springframework.http.ResponseEntity;
00008 import org.springframework.security.crypto.password.PasswordEncoder;
00009 import org.springframework.web.bind.annotation.*;
00010 import java.util.Optional;
00011 import org.slf4j.Logger;
00012 import org.slf4j.LoggerFactory;
00013
00021 @RestController
00022 @RequestMapping("/api")
00023 public class AuthController {
00024
00026   private static final Logger LOG = LoggerFactory.getLogger(AuthController.class);
00027
00029   @Autowired
00030   private UserRepository userRepository;
00031
00033   @Autowired
00034   private PasswordEncoder passwordEncoder;
00035
00042   public static class LoginRequest {
00043     public String username;
00044     public String password;
00045   }
00046
00060   @PostMapping("/login")
00061   public ResponseEntity<?> login(@RequestBody LoginRequest request) {
00062
00063     LOG.info("Login attempt for user: {}", request.username);
00064
00065     Optional<User> userOpt = userRepository.findByUsername(request.username);
00066
00067     if (!userOpt.isPresent()) {
00068       LOG.warn("User not found in DB: {}", request.username);
00069       return ResponseEntity.status(401).body("Invalid credentials");
00070     }
00071
00072     User user = userOpt.get();
00073     LOG.debug("User found. Stored hash = {}", user.getPasswordHash()); // Mejor debug, no info
00074
00075     if (!passwordEncoder.matches(request.password, user.getPasswordHash())) {
00076       LOG.warn("Password mismatch for user: {}", request.username);
00077       return ResponseEntity.status(401).body("Invalid credentials");
00078     }
00079
00080     LOG.info("Login OK for user: {}", request.username);
00081
00082     return ResponseEntity.ok(user);
00083
00084   }
00085
00086 }
```

10.22 CashierController.java File Reference

```
import es.ull.esit.server.middleware.model.Cashier;
import es.ull.esit.server.repo.CashierRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
```

```
import java.util.Optional;
```

Include dependency graph for CashierController.java:



Classes

- class es.ull.esit.server.controller.CashierController

  REST controller for managing cashiers.

Packages

- package es.ull.esit.server.controller

10.23 CashierController.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.controller;
00002
00003 import es.ull.esit.server.middleware.model.Cashier;
00004 import es.ull.esit.server.repo.CashierRepository;
00005 import org.springframework.beans.factory.annotation.Autowired;
00006 import org.springframework.http.HttpStatus;
00007 import org.springframework.http.ResponseEntity;
00008 import org.springframework.web.bind.annotation.*;
00009 import java.util.List;
00010 import java.util.Optional;
00011
00023 @RestController
00024 @RequestMapping("/api/cashiers")
00025 public class CashierController {
00026
00027   /* Repository used to perform database operations for Cashier entities. */
00028   @Autowired
00029   private CashierRepository cashierRepository;
00030
00039   @GetMapping
00040   public ResponseEntity<List<Cashier>> getAllCashiers() {
00041     List<Cashier> cashiers = cashierRepository.findAll();
00042     return ResponseEntity.ok(cashiers);
00043   }
00044
00054   @GetMapping("/{id}")
00055   public ResponseEntity<Cashier> getCashierById(@PathVariable Long id) {
00056     Optional<Cashier> cashier = cashierRepository.findById(id);
00057     return cashier
00058         .map(ResponseEntity::ok)
00059         .orElseGet(() -> ResponseEntity.notFound().build());
00060   }
00061
00071   @GetMapping("/name/{name}")
00072   public ResponseEntity<Cashier> getCashierByName(@PathVariable String name) {
00073     Optional<Cashier> cashier = cashierRepository.findByName(name);
00074     return cashier
00075         .map(ResponseEntity::ok)
00076         .orElseGet(() -> ResponseEntity.notFound().build());
00077   }
00078
00089   @PutMapping("/{id}")
00090   public ResponseEntity<Cashier> updateCashier(@PathVariable Long id,
00091       @RequestBody Cashier cashier) {
00092     Optional<Cashier> existingOpt = cashierRepository.findById(id);
00093     if (!existingOpt.isPresent()) {
00094       return ResponseEntity.notFound().build();
00095     }
00096
```

```
00097     Cashier existing = existingOpt.get();
00098     existing.setName(cashier.getName());
00099     existing.setSalary(cashier.getSalary());
00100
00101     Cashier updated = cashierRepository.save(existing);
00102     return ResponseEntity.ok(updated);
00103   }
00104
00105 }
```

10.24 DrinkController.java File Reference

```
import es.ull.esit.server.middleware.model.Drink;
import es.ull.esit.server.repo.DrinkRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;
```

Include dependency graph for DrinkController.java:

Classes

- class es.ull.esit.server.controller.DrinkController

  REST controller for managing drinks.

Packages

- package es.ull.esit.server.controller

10.25 DrinkController.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.controller;
00002
00003 import es.ull.esit.server.middleware.model.Drink;
00004 import es.ull.esit.server.repo.DrinkRepository;
00005 import org.springframework.beans.factory.annotation.Autowired;
00006 import org.springframework.http.HttpStatus;
00007 import org.springframework.http.ResponseEntity;
00008 import org.springframework.web.bind.annotation.*;
00009
00010 import java.util.List;
00011 import java.util.Optional;
00012
00020 @RestController
00021 @RequestMapping("/api/drinks")
00022 public class DrinkController {
00023
00025   @Autowired
00026   private DrinkRepository drinkRepository;
00027
00035   @GetMapping
```

```
00036   public ResponseEntity<List<Drink» getAllDrinks() {
00037     List<Drink> drinks = drinkRepository.findAll();
00038     return ResponseEntity.ok(drinks);
00039   }
00040
00049   @GetMapping("/{id}")
00050   public ResponseEntity<Drink> getDrinkById(@PathVariable Long id) {
00051     Optional<Drink> drink = drinkRepository.findById(id);
00052     return drink.map(ResponseEntity::ok)
00053         .orElseGet(() -> ResponseEntity.notFound().build());
00054   }
00055
00064   @PostMapping
00065   public ResponseEntity<Drink> createDrink(@RequestBody Drink drink) {
00066     Drink saved = drinkRepository.save(drink);
00067     return ResponseEntity.status(HttpStatus.CREATED).body(saved);
00068   }
00069
00079   @PutMapping("/{id}")
00080   public ResponseEntity<Drink> updateDrink(@PathVariable Long id,
00081       @RequestBody Drink drink) {
00082     if (!drinkRepository.existsById(id)) {
00083       return ResponseEntity.notFound().build();
00084     }
00085     drink.setDrinksId(id);
00086     Drink updated = drinkRepository.save(drink);
00087     return ResponseEntity.ok(updated);
00088   }
00089
00098   @DeleteMapping("/{id}")
00099   public ResponseEntity<Void> deleteDrink(@PathVariable Long id) {
00100     if (!drinkRepository.existsById(id)) {
00101       return ResponseEntity.notFound().build();
00102     }
00103     drinkRepository.deleteById(id);
00104     return ResponseEntity.noContent().build();
00105   }
00106 }
```

10.26 HealthController.java File Reference

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import javax.sql.DataSource;
import java.sql.Connection;
import java.util.HashMap;
import java.util.Map;
```

Include dependency graph for HealthController.java:

Classes

- class es.ull.esit.server.controller.HealthController

  REST controller for health and database connectivity checks.

Packages

- package es.ull.esit.server.controller

10.27 HealthController.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.controller;
00002
00003 import org.springframework.beans.factory.annotation.Autowired;
00004 import org.springframework.http.HttpStatus;
00005 import org.springframework.http.ResponseEntity;
00006 import org.springframework.web.bind.annotation.GetMapping;
00007 import org.springframework.web.bind.annotation.RequestMapping;
00008 import org.springframework.web.bind.annotation.RestController;
00009
00010 import javax.sql.DataSource;
00011 import java.sql.Connection;
00012 import java.util.HashMap;
00013 import java.util.Map;
00014
00025 @RestController
00026 @RequestMapping("/api")
00027 public class HealthController {
00028
00030   @Autowired
00031   private DataSource dataSource;
00032
00043   @GetMapping("/health")
00044   public ResponseEntity<Map<String, Object>> health() {
00045     Map<String, Object> response = new HashMap<>();
00046     response.put("status", "UP");
00047     response.put("timestamp", System.currentTimeMillis());
00048     response.put("service", "Restaurant Server");
00049     return ResponseEntity.ok(response);
00050   }
00051
00072   @GetMapping("/db-check")
00073   public ResponseEntity<Map<String, Object>> checkDatabase() {
00074     Map<String, Object> response = new HashMap<>();
00075
00076     try (Connection conn = dataSource.getConnection()) {
00077       boolean isValid = conn.isValid(2);
00078
00079       if (isValid) {
00080         response.put("status", "UP");
00081         response.put("database", "Connected");
00082         response.put("catalog", conn.getCatalog());
00083         response.put("url", conn.getMetaData().getURL());
00084         response.put("timestamp", System.currentTimeMillis());
00085         return ResponseEntity.ok(response);
00086       } else {
00087         response.put("status", "DOWN");
00088         response.put("database", "Connection not valid");
00089         return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE).body(response);
00090       }
00091     } catch (Exception e) {
00092       response.put("status", "DOWN");
00093       response.put("database", "Connection failed");
00094       response.put("error", e.getMessage());
00095       response.put("timestamp", System.currentTimeMillis());
00096       return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE).body(response);
00097     }
00098   }
00099 }
```

10.28 MainCourseController.java File Reference

```
import es.ull.esit.server.middleware.model.MainCourse;
import es.ull.esit.server.repo.MainCourseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;
```

Include dependency graph for MainCourseController.java:

Classes

- class es.ull.esit.server.controller.MainCourseController

  REST controller for managing main courses.

Packages

- package es.ull.esit.server.controller

10.29 MainCourseController.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.controller;
00002
00003 import es.ull.esit.server.middleware.model.MainCourse;
00004 import es.ull.esit.server.repo.MainCourseRepository;
00005 import org.springframework.beans.factory.annotation.Autowired;
00006 import org.springframework.http.*;
00007 import org.springframework.web.bind.annotation.*;
00008
00009 import java.util.List;
00010 import java.util.Optional;
00011
00020 @RestController
00021 @RequestMapping("/api/maincourses")
00022 public class MainCourseController {
00023
00025   @Autowired
00026   private MainCourseRepository mainCourseRepository;
00027
00036   @GetMapping
00037   public ResponseEntity<List<MainCourse» getAllMainCourses() {
00038     List<MainCourse> courses = mainCourseRepository.findAll();
00039     return ResponseEntity.ok(courses);
00040   }
00041
00051   @GetMapping("/{id}")
00052   public ResponseEntity<MainCourse> getMainCourseById(@PathVariable Long id) {
00053     Optional<MainCourse> course = mainCourseRepository.findById(id);
00054     return course
00055         .map(ResponseEntity::ok)
00056         .orElseGet(() -> ResponseEntity.notFound().build());
00057   }
00058
00067   @PostMapping
00068   public ResponseEntity<MainCourse> createMainCourse(@RequestBody MainCourse mainCourse) {
00069     MainCourse saved = mainCourseRepository.save(mainCourse);
00070     return ResponseEntity.status(HttpStatus.CREATED).body(saved);
00071   }
00072
00084   @PutMapping("/{id}")
00085   public ResponseEntity<MainCourse> updateMainCourse(@PathVariable Long id, @RequestBody MainCourse
      mainCourse) {
00086
00087     if (!mainCourseRepository.existsById(id)) {
00088       return ResponseEntity.notFound().build();
00089     }
00090
00091     mainCourse.setFoodId(id);
00092     MainCourse updated = mainCourseRepository.save(mainCourse);
00093     return ResponseEntity.ok(updated);
00094   }
```

```
00095
00104   @DeleteMapping("/{id}")
00105   public ResponseEntity<Void> deleteMainCourse(@PathVariable Long id) {
00106     if (!mainCourseRepository.existsById(id)) {
00107       return ResponseEntity.notFound().build();
00108     }
00109
00110     mainCourseRepository.deleteById(id);
00111     return ResponseEntity.noContent().build();
00112   }
00113 }
```

10.30 MenuController.java File Reference

```
import es.ull.esit.server.middleware.model.Appetizer;
import es.ull.esit.server.middleware.model.Drink;
import es.ull.esit.server.middleware.model.MainCourse;
import es.ull.esit.server.repo.AppetizerRepository;
import es.ull.esit.server.repo.DrinkRepository;
import es.ull.esit.server.repo.MainCourseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

Include dependency graph for MenuController.java:

Classes

- class es.ull.esit.server.controller.MenuController

  REST controller that exposes a consolidated restaurant menu.

Packages

- package es.ull.esit.server.controller

10.31 MenuController.java

Go to the documentation of this file.

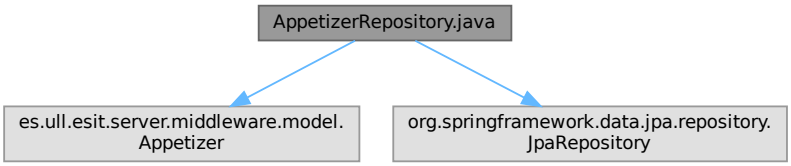
```
00001 package es.ull.esit.server.controller;
00002
00003 import es.ull.esit.server.middleware.model.Appetizer;
00004 import es.ull.esit.server.middleware.model.Drink;
00005 import es.ull.esit.server.middleware.model.MainCourse;
00006 import es.ull.esit.server.repo.AppetizerRepository;
00007 import es.ull.esit.server.repo.DrinkRepository;
00008 import es.ull.esit.server.repo.MainCourseRepository;
00009 import org.springframework.beans.factory.annotation.Autowired;
00010 import org.springframework.http.ResponseEntity;
00011 import org.springframework.web.bind.annotation.GetMapping;
```

```
00012 import org.springframework.web.bind.annotation.RequestMapping;
00013 import org.springframework.web.bind.annotation.RestController;
00014
00015 import java.util.HashMap;
00016 import java.util.List;
00017 import java.util.Map;
00018
00027 @RestController
00028 @RequestMapping("/api/menu")
00029 public class MenuController {
00030
00032   @Autowired
00033   private MainCourseRepository mainCourseRepository;
00034
00036   @Autowired
00037   private AppetizerRepository appetizerRepository;
00038
00040   @Autowired
00041   private DrinkRepository drinkRepository;
00042
00056   @GetMapping
00057   public ResponseEntity<Map<String, Object» getFullMenu() {
00058     Map<String, Object> menu = new HashMap<>();
00059
00060     List<MainCourse> mainCourses = mainCourseRepository.findAll();
00061     List<Appetizer> appetizers = appetizerRepository.findAll();
00062     List<Drink> drinks = drinkRepository.findAll();
00063
00064     menu.put("mainCourses", mainCourses);
00065     menu.put("appetizers", appetizers);
00066     menu.put("drinks", drinks);
00067     menu.put("totalItems", mainCourses.size() + appetizers.size() + drinks.size());
00068
00069     return ResponseEntity.ok(menu);
00070   }
00071 }
```

10.32 AppetizerRepository.java File Reference

```
import es.ull.esit.server.middleware.model.Appetizer;
import org.springframework.data.jpa.repository.JpaRepository;
```

Include dependency graph for AppetizerRepository.java:



Classes

- interface es.ull.esit.server.repo.AppetizerRepository

  Repository interface for managing appetizers in the database.

Packages

- package es.ull.esit.server.repo

10.33 AppetizerRepository.java

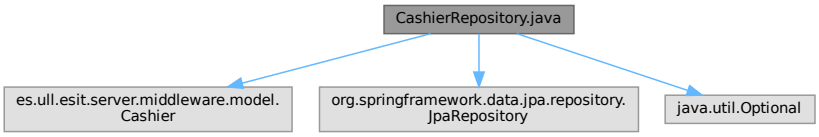
Go to the documentation of this file.

```
00001 package es.ull.esit.server.repo;
00002
00003 import es.ull.esit.server.middleware.model.Appetizer;
00004 import org.springframework.data.jpa.repository.JpaRepository;
00005
00012 public interface AppetizerRepository extends JpaRepository<Appetizer, Long> {
00013   // No extra methods are added for now, but custom queries can be defined here if
00014   // needed in the future.
00015 }
```

10.34 CashierRepository.java File Reference

```
import es.ull.esit.server.middleware.model.Cashier;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;
```

Include dependency graph for CashierRepository.java:



Classes

- interface es.ull.esit.server.repo.CashierRepository

  Repository interface for managing cashiers in the database.

Packages

- package es.ull.esit.server.repo

10.35 CashierRepository.java

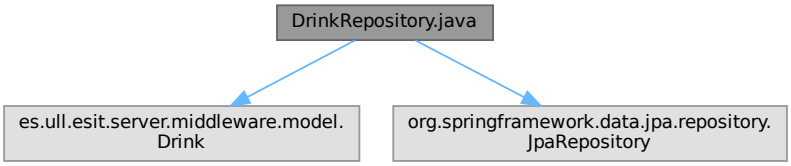
Go to the documentation of this file.

```
00001 package es.ull.esit.server.repo;
00002
00003 import es.ull.esit.server.middleware.model.Cashier;
00004 import org.springframework.data.jpa.repository.JpaRepository;
00005
00006 import java.util.Optional;
00007
00014 public interface CashierRepository extends JpaRepository<Cashier, Long> {
00015
00022   Optional<Cashier> findByName(String name);
00023 }
```

10.36 DrinkRepository.java File Reference

```
import es.ull.esit.server.middleware.model.Drink;
import org.springframework.data.jpa.repository.JpaRepository;
```

Include dependency graph for DrinkRepository.java:



Classes

- interface es.ull.esit.server.repo.DrinkRepository

  Repository interface for managing drinks in the database.

Packages

- package es.ull.esit.server.repo

10.37 DrinkRepository.java

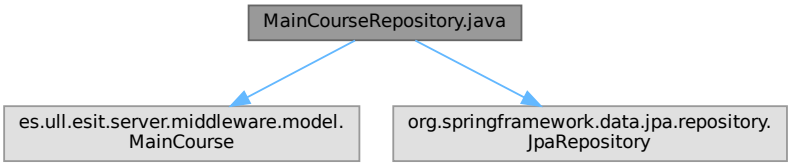
Go to the documentation of this file.

```
00001 package es.ull.esit.server.repo;
00002
00003 import es.ull.esit.server.middleware.model.Drink;
00004 import org.springframework.data.jpa.repository.JpaRepository;
00005
00012 public interface DrinkRepository extends JpaRepository<Drink, Long> {
00013   // No extra methods are added for now, but custom queries can be defined here if
00014   // needed in the future.
00015 }
```

10.38 MainCourseRepository.java File Reference

```
import es.ull.esit.server.middleware.model.MainCourse;
import org.springframework.data.jpa.repository.JpaRepository;
```

Include dependency graph for MainCourseRepository.java:



Classes

- interface es.ull.esit.server.repo.MainCourseRepository

  Repository interface for managing main courses in the database.

Packages

- package es.ull.esit.server.repo

10.39 MainCourseRepository.java

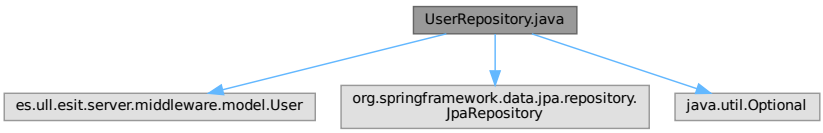
Go to the documentation of this file.

```
00001 package es.ull.esit.server.repo;
00002
00003 import es.ull.esit.server.middleware.model.MainCourse;
00004 import org.springframework.data.jpa.repository.JpaRepository;
00005
00013 public interface MainCourseRepository extends JpaRepository<MainCourse, Long> {
00014   // No extra methods are added for now, but custom queries can be defined here if
00015   // needed in the future.
00016 }
```

10.40 UserRepository.java File Reference

```
import es.ull.esit.server.middleware.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;
```

Include dependency graph for UserRepository.java:



Classes

- interface es.ull.esit.server.repo.UserRepository

  Repository interface for accessing users in the database.

Packages

- package es.ull.esit.server.repo

10.41 UserRepository.java

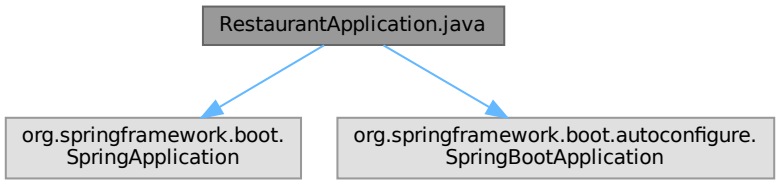
Go to the documentation of this file.

```
00001 package es.ull.esit.server.repo;
00002
00003 import es.ull.esit.server.middleware.model.User;
00004 import org.springframework.data.jpa.repository.JpaRepository;
00005 import java.util.Optional;
00006
00013 public interface UserRepository extends JpaRepository<User, Long> {
00020   Optional<User> findByUsername(String username);
00021 }
```

10.42 RestaurantApplication.java File Reference

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

Include dependency graph for RestaurantApplication.java:



Classes

- class es.ull.esit.server.RestaurantApplication

  Main Spring Boot application class for the restaurant backend.

Packages

- package es.ull.esit.server

10.43 RestaurantApplication.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server;
00002
00003 import org.springframework.boot.SpringApplication;
00004 import org.springframework.boot.autoconfigure.SpringBootApplication;
00005
00019 @SpringBootApplication
00020 public class RestaurantApplication {
00021
00030   public static void main(String[] args) {
00031     SpringApplication.run(RestaurantApplication.class, args);
00032   }
00033 }
```

10.44 AboutUs.java File Reference

Classes

- class es.ull.esit.app.AboutUs

  "About us" window displaying the restaurant's history and info.

Packages

- package es.ull.esit.app

10.45 AboutUs.java

Go to the documentation of this file.

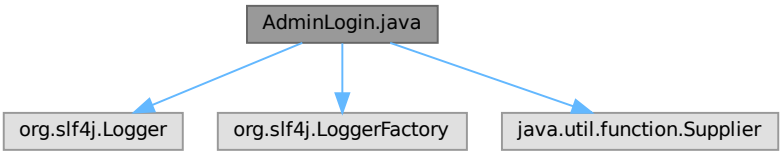
```
00001 package es.ull.esit.app;
00002
00014 public class AboutUs extends javax.swing.JFrame {
00015
00017   private String textBlock = """
00018       <html>
00019           <body>
00020               <tag>
00021                   <h1>Our Story</h1>
00022                   <p>Collecting flavors from all over the world to give everyone a taste of what they
      love.</p>
00023                   <p>Black Plate came in with this vision in mind, to be more of a home rather than an
      establishment.</p>
00024                   <p>2021 marked the beginning of the journey of Black Plate, starting from Al
      Khobar.</p>
00025                   <p>Because your visit means a lot to us, we would love to hear your suggestions and
      concerns so we can improve!</p>
00026               </tag>
00027           </body>
00028       </html>""";
00029
00035   public AboutUs() {
00036     initComponents();
00037   }
00038
00050   @SuppressWarnings("unchecked")
00051   // <editor-fold defaultstate="collapsed" desc="Generated
00052   // Code">//GEN-BEGIN:initComponents
00053   private void initComponents() {
00054
00055     jPanel1 = new javax.swing.JPanel();
00056     jButton3 = new javax.swing.JButton();
00057     jScrollPane1 = new javax.swing.JScrollPane();
00058     ourStory = new javax.swing.JTextArea();
00059     jLabel1 = new javax.swing.JLabel();
00060
00061     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00062     setTitle("Info");
00063     setResizable(false);
00064
00065     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00066
00067     jButton3.setBackground(new java.awt.Color(255, 255, 255));
00068     jButton3.setFont(new java.awt.Font("Yu Gothic UI", 1, 18)); // NOI18N
00069     jButton3.setText("Go Back");
00070     jButton3.addActionListener(this::jButton3ActionPerformed);
00071
00072     ourStory.setEditable(false);
00073     ourStory.setBackground(new java.awt.Color(248, 244, 230));
00074     ourStory.setColumns(20);
00075     ourStory.setFont(new java.awt.Font("Yu Gothic UI Light", 0, 14)); // NOI18N
00076     ourStory.setRows(5);
00077     ourStory.setText(textBlock);
00078     ourStory.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));
00079     jScrollPane1.setViewportView(ourStory);
00080
00081     // Updated path to match other UI resources
00082     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png")));
00083
```

```
00084     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00085     jPanel1.setLayout(jPanel1Layout);
00086     jPanel1Layout.setHorizontalGroup(
00087         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00088             .addGroup(jPanel1Layout.createSequentialGroup()
00089                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00090                     .addGroup(jPanel1Layout.createSequentialGroup()
00091                         .addGap(55, 55, 55)
00092 
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00093                             .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00094                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00095                             .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 1004,
00096                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00097                     .addGroup(jPanel1Layout.createSequentialGroup()
00098                         .addGap(489, 489, 489)
00099                         .addComponent(jLabel1)))
00100                 .addContainerGap(159, Short.MAX_VALUE)));
00101     jPanel1Layout.setVerticalGroup(
00102         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00103             .addGroup(jPanel1Layout.createSequentialGroup()
00104                 .addContainerGap(71, Short.MAX_VALUE)
00105                 .addComponent(jLabel1)
00106                 .addGap(67, 67, 67)
00107                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00108                     javax.swing.GroupLayout.PREFERRED_SIZE)
00109                 .addGap(26, 26, 26)
00110                 .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00111                     javax.swing.GroupLayout.PREFERRED_SIZE)
00112                 .addGap(30, 30, 30)));
00113 
00114     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00115     getContentPane().setLayout(layout);
00116     layout.setHorizontalGroup(
00117         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00118             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00119                 Short.MAX_VALUE));
00120     layout.setVerticalGroup(
00121         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00122             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00123                 Short.MAX_VALUE));
00124 
00125     pack();
00126     setLocationRelativeTo(null);
00127   }// </editor-fold>//GEN-END:initComponents
00128 
00137   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton3ActionPerformed
00138     new CashierLogin().setVisible(true);
00139     this.dispose(); // Close current window
00140   }// GEN-LAST:event_jButton3ActionPerformed
00141 
00152   public static void main(String[] args) {
00153     try {
00154       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00155         if ("Nimbus".equals(info.getName())) {
00156           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00157           break;
00158         }
00159       }
00160     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00161         | javax.swing.UnsupportedLookAndFeelException ex) {
00162       java.util.logging.Logger.getLogger(AboutUs.class.getName())
00163           .log(java.util.logging.Level.SEVERE, null, ex);
00164     }
00165     // </editor-fold>
00166 
00167     /* Create and display the form */
00168     java.awt.EventQueue.invokeLater(() -> new AboutUs().setVisible(true));
00169   }
00170 
00171   // Variables declaration - do not modify//GEN-BEGIN:variables
00172   // Sonarqube rule java:S1450 must be ignored here as these variables are
00173   // auto-generated by the Form Editor and need to remain as instance variables.
00175   private javax.swing.JButton jButton3;
00177   private javax.swing.JLabel jLabel1;
00179   private javax.swing.JPanel jPanel1;
00181   private javax.swing.JScrollPane jScrollPane1;
00183   private javax.swing.JTextArea ourStory;
00184   // End of variables declaration//GEN-END:variables
00185 }
```

10.46 AdminLogin.java File Reference

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.function.Supplier;
```

Include dependency graph for AdminLogin.java:



Classes

- class es.ull.esit.app.AdminLogin

  Login window for authenticating administrators.

Packages

- package es.ull.esit.app

10.47 AdminLogin.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app;
00002
00003 import org.slf4j.Logger;
00004 import org.slf4j.LoggerFactory;
00005 import java.util.function.Supplier;
00006
00016 public class AdminLogin extends javax.swing.JFrame {
00017
00019   private static final String PRIMARY_FONT = "Yu Gothic UI";
00020
00022   private static final Logger LOGGER = LoggerFactory.getLogger(AdminLogin.class);
00023
00024     /*
00025      * Package-private suppliers used to create dependent windows. Tests may
00026      * override these to inject lightweight stubs without needing to shadow
00027      * the production classes. They default to creating the real windows.
00028      */
00029     static Supplier<java.awt.Window> adminProductsSupplier = () -> new AdminProducts();
00030     static Supplier<java.awt.Window> orderSupplier = () -> new Order();
00031     static Supplier<java.awt.Window> loginSupplier = () -> new Login();
00032
00038   public AdminLogin() {
00039     initComponents();
00040   }
00041
00049   @SuppressWarnings("unchecked")
00050   // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
00051   private void initComponents() {
00052
00053     jPanel1 = new javax.swing.JPanel();
00054     jLabel3 = new javax.swing.JLabel();
00055     jLabel1 = new javax.swing.JLabel();
```

```
00056     jButton2 = new javax.swing.JButton();
00057     jButton1 = new javax.swing.JButton();
00058     jButton3 = new javax.swing.JButton();
00059 
00060     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00061     setTitle("Admin ");
00062     setResizable(false);
00063 
00064     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00065 
00066     jLabel3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 24)); // NOI18N
00067     jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00068     jLabel3.setText("Welcome Admin");
00069 
00070     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
      NOI18N
00071 
00072     jButton2.setBackground(new java.awt.Color(153, 153, 153));
00073     jButton2.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00074     jButton2.setText("Update Prices");
00075     jButton2.addActionListener(this::jButton2ActionPerformed);
00076 
00077     jButton1.setBackground(new java.awt.Color(153, 153, 153));
00078     jButton1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00079     jButton1.setText("Menu");
00080     jButton1.addActionListener(this::jButton1ActionPerformed);
00081 
00082     jButton3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00083     jButton3.setText("LogOut");
00084     jButton3.addActionListener(this::jButton3ActionPerformed);
00085 
00086     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00087     jPanel1.setLayout(jPanel1Layout);
00088     jPanel1Layout.setHorizontalGroup(
00089         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00090             .addGroup(jPanel1Layout.createSequentialGroup()
00091                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00092                     .addGroup(jPanel1Layout.createSequentialGroup()
00093                         .addGap(140, 140, 140)
00094                         .addComponent(jLabel1))
00095                     .addGroup(jPanel1Layout.createSequentialGroup()
00096                         .addGap(66, 66, 66)
00097 
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00098                             .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00099                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00100                             .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00101                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00102                             .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 306,
00103                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00104                     .addGroup(jPanel1Layout.createSequentialGroup()
00105                         .addGap(152, 152, 152)
00106                         .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00107                             javax.swing.GroupLayout.PREFERRED_SIZE)))
00108                 .addContainerGap(69, Short.MAX_VALUE)));
00109     jPanel1Layout.setVerticalGroup(
00110         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00111             .addGroup(jPanel1Layout.createSequentialGroup()
00112                 .addGap(31, 31, 31)
00113                 .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
00114                     javax.swing.GroupLayout.PREFERRED_SIZE)
00115                 .addGap(18, 18, 18)
00116                 .addComponent(jLabel1)
00117                 .addGap(29, 29, 29)
00118                 .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00119                     javax.swing.GroupLayout.PREFERRED_SIZE)
00120                 .addGap(18, 18, 18)
00121                 .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00122                     javax.swing.GroupLayout.PREFERRED_SIZE)
00123                 .addGap(29, 29, 29)
00124                 .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00125                     javax.swing.GroupLayout.PREFERRED_SIZE)
00126                 .addContainerGap(115, Short.MAX_VALUE)));
00127 
00128     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00129     getContentPane().setLayout(layout);
00130     layout.setHorizontalGroup(
00131         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00132             .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00133                 javax.swing.GroupLayout.PREFERRED_SIZE));
00134     layout.setVerticalGroup(
00135         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00136             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00137                 Short.MAX_VALUE));
00138 
```

```
00139     pack();
00140     setLocationRelativeTo(null);
00141   }// </editor-fold>//GEN-END:initComponents
00142
00153   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton2ActionPerformed
00154   try {
00155     adminProductsSupplier.get().setVisible(true);
00156     this.dispose();
00157   } catch (Exception ex) {
00158         LOGGER.error("Error opening product admin window", ex);
00159         javax.swing.JOptionPane.showMessageDialog(
00160             this,
00161             "Error opening product admin window:\n" + ex.getMessage(),
00162             "Error",
00163             javax.swing.JOptionPane.ERROR_MESSAGE
00164         );
00165     }
00166   }// GEN-LAST:event_jButton2ActionPerformed
00167
00176   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton1ActionPerformed
00177   try {
00178     orderSupplier.get().setVisible(true);
00179     this.dispose();
00180   } catch (Exception ex) {
00181         LOGGER.error("Error opening menu window", ex);
00182         javax.swing.JOptionPane.showMessageDialog(
00183             this,
00184             "Error opening menu window:\n" + ex.getMessage(),
00185             "Error",
00186             javax.swing.JOptionPane.ERROR_MESSAGE
00187         );
00188     }
00189   }// GEN-LAST:event_jButton1ActionPerformed
00190
00199   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton3ActionPerformed
00200   loginSupplier.get().setVisible(true);
00201   this.dispose();
00202   }// GEN-LAST:event_jButton3ActionPerformed
00203
00213   public static void main(String[] args) {
00214     try {
00215       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00216         if ("Nimbus".equals(info.getName())) {
00217           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00218           break;
00219         }
00220       }
00221     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00222         | javax.swing.UnsupportedLookAndFeelException ex) {
00223
      java.util.logging.Logger.getLogger(AdminLogin.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00224     }
00225     // </editor-fold>
00226
00227     java.awt.EventQueue.invokeLater(() -> new AdminLogin().setVisible(true));
00228   }
00229
00230   // Variables declaration - do not modify//GEN-BEGIN:variables
00231   // Sonarqube rule java:S1450 must be ignored here as these variables are
00232   // auto-generated by the Form Editor and need to remain as instance variables.
00234   private javax.swing.JButton jButton1;
00236   private javax.swing.JButton jButton2;
00238   private javax.swing.JButton jButton3;
00240   private javax.swing.JLabel jLabel1;
00242   private javax.swing.JLabel jLabel3;
00244   private javax.swing.JPanel jPanel1;
00245   // End of variables declaration//GEN-END:variables
00246 }
```

10.48 AdminProducts.java File Reference

```
import es.ull.esit.app.middleware.ApiClient;
import es.ull.esit.app.middleware.model.Appetizer;
import es.ull.esit.app.middleware.model.Drink;
import es.ull.esit.app.middleware.model.MainCourse;
```

```
import es.ull.esit.app.middleware.service.ProductService;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableModel;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

Include dependency graph for AdminProducts.java:

Classes

- class es.ull.esit.app.AdminProducts

  Administrative window for managing products and prices.

Packages

- package es.ull.esit.app

10.49 AdminProducts.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app;
00002
00003 import es.ull.esit.app.middleware.ApiClient;
00004 import es.ull.esit.app.middleware.model.Appetizer;
00005 import es.ull.esit.app.middleware.model.Drink;
00006 import es.ull.esit.app.middleware.model.MainCourse;
00007 import es.ull.esit.app.middleware.service.ProductService;
00008 import java.util.List;
00009 import javax.swing.JOptionPane;
00010 import javax.swing.SwingUtilities;
00011 import javax.swing.table.DefaultTableModel;
00012 import org.slf4j.Logger;
00013 import org.slf4j.LoggerFactory;
00014
00015
00027 public class AdminProducts extends javax.swing.JFrame {
00028
00030   private final transient ProductService productService;
00031
00033   private static final Logger LOGGER = LoggerFactory.getLogger(AdminProducts.class);
00034
00036   private static final String PRIMARY_FONT = "Yu Gothic UI";
00038   private static final String SECONDARY_FONT = "Thonburi";
00039
00041   private static final String FIRST_COLUMN_HEADER = "ID";
00043   private static final String SECOND_COLUMN_HEADER = "Item Name";
00045   private static final String THIRD_COLUMN_HEADER = "Item Price";
00046
00048   private static final String UPDATE_STRING = "Update";
00049
00051   DefaultTableModel modelDrink;
00053   DefaultTableModel modelappetizers;
00055   DefaultTableModel modelmaincourse;
00056
00058   String[] columnNames = { FIRST_COLUMN_HEADER, SECOND_COLUMN_HEADER, THIRD_COLUMN_HEADER };
00059
00061   Long selectedDrinkID;
00063   Long selectedAppetizerID;
00065   Long selectedMainCourseID;
00066
00074   public AdminProducts() {
```

```
00075     initComponents();
00076
00077     // Initialize the Service Layer.
00078     ApiClient client = new ApiClient("http://localhost:8080");
00079     this.productService = new ProductService(client);
00080
00081     // Initialize table models and set shared headers.
00082     modelDrink = new DefaultTableModel();
00083     modelDrink.setColumnIdentifiers(columnNames);
00084     modelappetizers = new DefaultTableModel();
00085     modelappetizers.setColumnIdentifiers(columnNames);
00086     modelmaincourse = new DefaultTableModel();
00087     modelmaincourse.setColumnIdentifiers(columnNames);
00088
00089     // Link models to tables.
00090     jTable1.setModel(modelDrink);
00091     jTable2.setModel(modelappetizers);
00092     jTable3.setModel(modelmaincourse);
00093
00094     // Load initial data from backend.
00095     refreshAllTables();
00096   }
00097
00105   AdminProducts(ProductService productService, boolean startLoading) {
00106     initComponents();
00107
00108     // Use injected service instead of creating a new ApiClient.
00109     this.productService = productService;
00110
00111     // Initialize table models and set shared headers.
00112     modelDrink = new DefaultTableModel();
00113     modelDrink.setColumnIdentifiers(columnNames);
00114     modelappetizers = new DefaultTableModel();
00115     modelappetizers.setColumnIdentifiers(columnNames);
00116     modelmaincourse = new DefaultTableModel();
00117     modelmaincourse.setColumnIdentifiers(columnNames);
00118
00119     // Link models to tables.
00120     jTable1.setModel(modelDrink);
00121     jTable2.setModel(modelappetizers);
00122     jTable3.setModel(modelmaincourse);
00123
00124     if (startLoading) {
00125       refreshAllTables();
00126     }
00127   }
00128
00135   private void refreshAllTables() {
00136     loadDrinks();
00137     loadAppetizer();
00138     loadmainCourse();
00139   }
00140
00148   void loadDrinks() {
00149     new Thread(() -> {
00150       try {
00151         List<Drink> drinks = productService.getAllDrinks();
00152         SwingUtilities.invokeLater(() -> {
00153           modelDrink.setRowCount(0);
00154           for (Drink drink : drinks) {
00155             modelDrink.addRow(new Object[] {
00156                 drink.getDrinksId(),
00157                 drink.getItemDrinks(),
00158                 drink.getDrinksPrice()
00159             });
00160           }
00161         });
00162       } catch (Exception ex) {
00163         SwingUtilities
00164             .invokeLater(() -> JOptionPane.showMessageDialog(null, "Error loading drinks: " +
      ex.getMessage()));
00165       }
00166     }).start();
00167   }
00168
00174   void loadDrinksSync() {
00175     try {
00176       List<Drink> drinks = productService.getAllDrinks();
00177       modelDrink.setRowCount(0);
00178       for (Drink drink : drinks) {
00179         modelDrink.addRow(new Object[] { drink.getDrinksId(), drink.getItemDrinks(),
      drink.getDrinksPrice() });
00180       }
00181     } catch (Exception ex) {
00182       // In tests we prefer to propagate exceptions; keep same behaviour as async but avoid dialogs.
00183       throw new RuntimeException(ex);
00184     }
```

```
00185   }
00186
00193   void loadAppetizer() {
00194     new Thread(() -> {
00195       try {
00196         List<Appetizer> appetizers = productService.getAllAppetizers();
00197         SwingUtilities.invokeLater(() -> {
00198           modelappetizers.setRowCount(0);
00199           for (Appetizer appetizer : appetizers) {
00200             modelappetizers.addRow(new Object[] {
00201                 appetizer.getAppetizersId(),
00202                 appetizer.getItemAppetizers(),
00203                 appetizer.getAppetizersPrice()
00204             });
00205           }
00206         });
00207       } catch (Exception ex) {
00208         SwingUtilities
00209             .invokeLater(() -> JOptionPane.showMessageDialog(null, "Error loading appetizers: " +
      ex.getMessage()));
00210       }
00211     }).start();
00212   }
00213
00217   void loadAppetizerSync() {
00218     try {
00219       List<Appetizer> appetizers = productService.getAllAppetizers();
00220       modelappetizers.setRowCount(0);
00221       for (Appetizer appetizer : appetizers) {
00222         modelappetizers.addRow(new Object[] { appetizer.getAppetizersId(),
      appetizer.getItemAppetizers(), appetizer.getAppetizersPrice() });
00223       }
00224     } catch (Exception ex) {
00225       throw new RuntimeException(ex);
00226     }
00227   }
00228
00235   void loadmainCourse() {
00236     new Thread(() -> {
00237       try {
00238         List<MainCourse> mainCourses = productService.getAllMainCourses();
00239         SwingUtilities.invokeLater(() -> {
00240           modelmaincourse.setRowCount(0);
00241           for (MainCourse mainCourse : mainCourses) {
00242             modelmaincourse.addRow(new Object[] {
00243                 mainCourse.getFoodId(),
00244                 mainCourse.getItemFood(),
00245                 mainCourse.getFoodPrice()
00246             });
00247           }
00248         });
00249       } catch (Exception ex) {
00250         SwingUtilities
00251             .invokeLater(() -> JOptionPane.showMessageDialog(null, "Error loading main courses: " +
      ex.getMessage()));
00252       }
00253     }).start();
00254   }
00255
00259   void loadmainCourseSync() {
00260     try {
00261       List<MainCourse> mainCourses = productService.getAllMainCourses();
00262       modelmaincourse.setRowCount(0);
00263       for (MainCourse mainCourse : mainCourses) {
00264         modelmaincourse.addRow(new Object[] { mainCourse.getFoodId(), mainCourse.getItemFood(),
      mainCourse.getFoodPrice() });
00265       }
00266     } catch (Exception ex) {
00267       throw new RuntimeException(ex);
00268     }
00269   }
00270
00277   void selectDrinkByRowSync(int row) {
00278     if (row < 0 || row >= modelDrink.getRowCount())
00279       return;
00280     String idStr = modelDrink.getValueAt(row, 0).toString();
00281     Long id = Long.valueOf(idStr);
00282     Drink drink = productService.getDrinkById(id);
00283     itemname.setText(drink.getItemDrinks());
00284     itemprice.setText(String.valueOf(drink.getDrinksPrice()));
00285   }
00286
00287   void selectAppetizerByRowSync(int row) {
00288     if (row < 0 || row >= modelappetizers.getRowCount())
00289       return;
00290     String idStr = modelappetizers.getValueAt(row, 0).toString();
00291     Long id = Long.valueOf(idStr);
```

```
00292     Appetizer appetizer = productService.getAppetizerById(id);
00293     itemname1.setText(appetizer.getItemAppetizers());
00294     itemprice1.setText(String.valueOf(appetizer.getAppetizersPrice()));
00295   }
00296 
00297   void selectMainCourseByRowSync(int row) {
00298     if (row < 0 || row >= modelmaincourse.getRowCount())
00299       return;
00300     String idStr = modelmaincourse.getValueAt(row, 0).toString();
00301     Long id = Long.valueOf(idStr);
00302     MainCourse mainCourse = productService.getMainCourseById(id);
00303     itemname2.setText(mainCourse.getItemFood());
00304     itemprice2.setText(String.valueOf(mainCourse.getFoodPrice()));
00305   }
00306 
00315   @SuppressWarnings("unchecked")
00316   // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
00317   private void initComponents() {
00318 
00319     jPanel1 = new javax.swing.JPanel();
00320     jLabel1 = new javax.swing.JLabel();
00321     jTabbedPane1 = new javax.swing.JTabbedPane();
00322     jPanel2 = new javax.swing.JPanel();
00323     itemname = new javax.swing.JTextField();
00324     itemprice = new javax.swing.JTextField();
00325     jScrollPane1 = new javax.swing.JScrollPane();
00326     jTable1 = new javax.swing.JTable();
00327     jButton1 = new javax.swing.JButton();
00328     jButton2 = new javax.swing.JButton();
00329     jLabel2 = new javax.swing.JLabel();
00330     jLabel3 = new javax.swing.JLabel();
00331     jLabel4 = new javax.swing.JLabel();
00332     jPanel3 = new javax.swing.JPanel();
00333     jLabel5 = new javax.swing.JLabel();
00334     jLabel6 = new javax.swing.JLabel();
00335     jLabel7 = new javax.swing.JLabel();
00336     jScrollPane2 = new javax.swing.JScrollPane();
00337     jTable2 = new javax.swing.JTable();
00338     itemname1 = new javax.swing.JTextField();
00339     itemprice1 = new javax.swing.JTextField();
00340     jButton4 = new javax.swing.JButton();
00341     jButton5 = new javax.swing.JButton();
00342     jPanel4 = new javax.swing.JPanel();
00343     jLabel8 = new javax.swing.JLabel();
00344     jLabel9 = new javax.swing.JLabel();
00345     jLabel10 = new javax.swing.JLabel();
00346     jScrollPane3 = new javax.swing.JScrollPane();
00347     jTable3 = new javax.swing.JTable();
00348     itemname2 = new javax.swing.JTextField();
00349     itemprice2 = new javax.swing.JTextField();
00350     jButton6 = new javax.swing.JButton();
00351     jButton7 = new javax.swing.JButton();
00352     jButton3 = new javax.swing.JButton();
00353 
00354     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00355     setTitle("Admin");
00356     setResizable(false);
00357 
00358     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00359 
00360     jLabel1.setBackground(new java.awt.Color(255, 153, 0));
00361     jLabel1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 36)); // NOI18N
00362     jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00363     jLabel1.setText("Items Prices Update Portal");
00364 
00365     jTabbedPane1.setBackground(new java.awt.Color(248, 244, 230));
00366     jTabbedPane1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true));
00367     jTabbedPane1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
00368     jTabbedPane1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00369 
00370     jPanel2.setBackground(new java.awt.Color(248, 244, 230));
00371 
00372     itemname.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00373     itemname.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00374     itemname.setBorder(javax.swing.BorderFactory.createTitledBorder(
00375         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), SECOND_COLUMN_HEADER,
00376         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00377         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00378 
00379     itemprice.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00380     itemprice.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00381     itemprice.setBorder(javax.swing.BorderFactory.createTitledBorder(
00382         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), THIRD_COLUMN_HEADER,
00383         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00384         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00385 
00386     jTable1.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
```

```
00387     jTable1.setModel(new javax.swing.table.DefaultTableModel(
00388         new Object[][] {
00389             {},
00390             {},
00391             {},
00392             {}
00393         },
00394         new String[] {
00395 
00396         }));
00397     jTable1.setInheritsPopupMenu(true);
00398     jTable1.getTableHeader().setResizingAllowed(false);
00399     jTable1.getTableHeader().setReorderingAllowed(false);
00400     jTable1.addMouseListener(new java.awt.event.MouseAdapter() {
00401       @Override
00402       public void mouseClicked(java.awt.event.MouseEvent evt) {
00403         jTable1MouseClicked(evt);
00404       }
00405     });
00406     jTable1.addKeyListener(new java.awt.event.KeyAdapter() {
00407       @Override
00408       public void keyPressed(java.awt.event.KeyEvent evt) {
00409         jTable1KeyPressed(evt);
00410       }
00411     });
00412     jScrollPane1.setViewportView(jTable1);
00413 
00414     jButton1.setBackground(new java.awt.Color(255, 255, 255));
00415     jButton1.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00416     jButton1.setText(UPDATE_STRING);
00417     jButton1.addActionListener(this::jButton1ActionPerformed);
00418 
00419     jButton2.setBackground(new java.awt.Color(255, 255, 255));
00420     jButton2.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00421     jButton2.setText("Add");
00422     jButton2.addActionListener(this::jButton2ActionPerformed);
00423 
00424     jLabel2.setBackground(new java.awt.Color(255, 153, 0));
00425     jLabel2.setFont(new java.awt.Font(PRIMARY_FONT, 1, 25)); // NOI18N
00426     jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00427     jLabel2.setText("Available Drinks");
00428 
00429     jLabel3.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00430     jLabel3.setText("Enter new drink information carefully to add.");
00431 
00432     jLabel4.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00433     jLabel4.setText("Please select the drink to update the price.");
00434 
00435     javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
00436     jPanel2.setLayout(jPanel2Layout);
00437     jPanel2Layout.setHorizontalGroup(
00438         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00439             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel2Layout.createSequentialGroup()
00440                 .addGap(0, 27, Short.MAX_VALUE)
00441                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00442                     .addGroup(jPanel2Layout.createSequentialGroup()
00443                         .addGap(24, 24, 24)
00444 
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00445                             .addGroup(jPanel2Layout.createSequentialGroup()
00446                                 .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
00447                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00448                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
      109,
00449                                     Short.MAX_VALUE)
00450                                 .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
00451                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00452                                 .addGap(8, 8, 8))
00453                             .addComponent(itemprice)
00454                             .addComponent(itemname, javax.swing.GroupLayout.DEFAULT_SIZE, 311,
      Short.MAX_VALUE)
00455                             .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
00456                                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00457                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00458                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00459 
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00460                             .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 586,
00461                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00462                             .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 408,
00463                                 javax.swing.GroupLayout.PREFERRED_SIZE))
00464                         .addGap(115, 115, 115))
00465                     .addGroup(jPanel2Layout.createSequentialGroup()
00466                         .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 1111,
00467                             javax.swing.GroupLayout.PREFERRED_SIZE)
00468                         .addGap(18, 18, 18)))));
```

```
00469     jPanel2Layout.setVerticalGroup(
00470         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00471             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel2Layout.createSequentialGroup()
00472                 .addContainerGap()
00473                 .addComponent(jLabel2)
00474                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
      Short.MAX_VALUE)
00475
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00476                     .addComponent(jLabel3)
00477                     .addComponent(jLabel4))
00478                 .addGap(18, 18, 18)
00479                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00480                     .addGroup(jPanel2Layout.createSequentialGroup()
00481                         .addComponent(itemname, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00482                             javax.swing.GroupLayout.PREFERRED_SIZE)
00483                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00484                         .addComponent(itemprice, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00485                             javax.swing.GroupLayout.PREFERRED_SIZE)
00486                         .addGap(37, 37, 37)
00487
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00488                             .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00489                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00490                             .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00491                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00492                     .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00493                         javax.swing.GroupLayout.PREFERRED_SIZE))
00494                 .addGap(48, 48, 48)));
00495
00496     jTabbedPane1.addTab("Drinks", jPanel2);
00497
00498     jPanel3.setBackground(new java.awt.Color(248, 244, 230));
00499
00500     jLabel5.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00501     jLabel5.setText("Enter new appetizer information carefully to add.");
00502
00503     jLabel6.setBackground(new java.awt.Color(255, 153, 0));
00504     jLabel6.setFont(new java.awt.Font(PRIMARY_FONT, 1, 25)); // NOI18N
00505     jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00506     jLabel6.setText("Available Appetizer");
00507
00508     jLabel7.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00509     jLabel7.setText("Please select the appetizer to update the price.");
00510
00511     jTable2.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00512     jTable2.setModel(new javax.swing.table.DefaultTableModel(
00513         new Object[][] {
00514             {},
00515             {},
00516             {},
00517             {}
00518         },
00519         new String[] {
00520
00521         }));
00522     jTable2.addMouseListener(new java.awt.event.MouseAdapter() {
00523       @Override
00524       public void mouseClicked(java.awt.event.MouseEvent evt) {
00525         jTable2MouseClicked(evt);
00526       }
00527     });
00528     jScrollPane2.setViewportView(jTable2);
00529
00530     itemname1.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00531     itemname1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00532     itemname1.setBorder(javax.swing.BorderFactory.createTitledBorder(
00533         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), SECOND_COLUMN_HEADER,
00534         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00535         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00536
00537     itemprice1.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00538     itemprice1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00539     itemprice1.setBorder(javax.swing.BorderFactory.createTitledBorder(
00540         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), THIRD_COLUMN_HEADER,
00541         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00542         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00543
00544     jButton4.setBackground(new java.awt.Color(255, 255, 255));
00545     jButton4.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00546     jButton4.setText(UPDATE_STRING);
00547     jButton4.addActionListener(this::jButton4ActionPerformed);
00548
00549     jButton5.setBackground(new java.awt.Color(255, 255, 255));
00550     jButton5.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00551     jButton5.setText("Add");
```

```
00552     jButton5.addActionListener(this::jButton5ActionPerformed);
00553
00554     javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
00555     jPanel3.setLayout(jPanel3Layout);
00556     jPanel3Layout.setHorizontalGroup(
00557         jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00558             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel3Layout.createSequentialGroup()
00559                 .addGap(0, 27, Short.MAX_VALUE)
00560                 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00561                     .addGroup(jPanel3Layout.createSequentialGroup()
00562                         .addGap(24, 24, 24)
00563
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00564                             .addGroup(jPanel3Layout.createSequentialGroup()
00565                                 .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
00566                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00567                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
00568                                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00569                                 .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
00570                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00571                                 .addGap(8, 8, 8))
00572                             .addComponent(itemprice1)
00573                             .addComponent(itemname1)
00574                             .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE,
00575                                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00576                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00577                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00578
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00579                             .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 586,
00580                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00581                             .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 408,
00582                                 javax.swing.GroupLayout.PREFERRED_SIZE))
00583                         .addGap(115, 115, 115))
00584                     .addGroup(jPanel3Layout.createSequentialGroup()
00585                         .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 1111,
00586                             javax.swing.GroupLayout.PREFERRED_SIZE)
00587                         .addGap(18, 18, 18)))));
00588     jPanel3Layout.setVerticalGroup(
00589         jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00590             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel3Layout.createSequentialGroup()
00591                 .addContainerGap()
00592                 .addComponent(jLabel6)
00593                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
      Short.MAX_VALUE)
00594
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00595                     .addComponent(jLabel5)
00596                     .addComponent(jLabel7))
00597                 .addGap(18, 18, 18)
00598                 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00599                     .addGroup(jPanel3Layout.createSequentialGroup()
00600                         .addComponent(itemname1, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00601                             javax.swing.GroupLayout.PREFERRED_SIZE)
00602                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00603                         .addComponent(itemprice1, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00604                             javax.swing.GroupLayout.PREFERRED_SIZE)
00605                         .addGap(37, 37, 37)
00606
      .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00607                             .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00608                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00609                             .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00610                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00611                     .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00612                         javax.swing.GroupLayout.PREFERRED_SIZE))
00613                 .addGap(48, 48, 48)));
00614
00615     jTabbedPane1.addTab("Appetizers", jPanel3);
00616
00617     jPanel4.setBackground(new java.awt.Color(248, 244, 230));
00618     jPanel4.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00619
00620     jLabel8.setBackground(new java.awt.Color(248, 244, 230));
00621     jLabel8.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00622     jLabel8.setText("Enter new item information carefully to add.");
00623
00624     jLabel9.setBackground(new java.awt.Color(255, 153, 0));
00625     jLabel9.setFont(new java.awt.Font(PRIMARY_FONT, 1, 25)); // NOI18N
00626     jLabel9.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00627     jLabel9.setText("Avaliable MainCourse");
00628
00629     jLabel10.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00630     jLabel10.setText("Please select the item to update the price.");
00631
```

```
00632     jTable3.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00633     jTable3.setModel(new javax.swing.table.DefaultTableModel(
00634         new Object[][] {
00635             {},
00636             {},
00637             {},
00638             {}
00639         },
00640         new String[] {
00641 
00642         }));
00643     jTable3.addMouseListener(new java.awt.event.MouseAdapter() {
00644       @Override
00645       public void mouseClicked(java.awt.event.MouseEvent evt) {
00646         jTable3MouseClicked(evt);
00647       }
00648     });
00649     jScrollPane3.setViewportView(jTable3);
00650 
00651     itemname2.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00652     itemname2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00653     itemname2.setBorder(javax.swing.BorderFactory.createTitledBorder(
00654         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), SECOND_COLUMN_HEADER,
00655         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00656         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00657 
00658     itemprice2.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00659     itemprice2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00660     itemprice2.setBorder(javax.swing.BorderFactory.createTitledBorder(
00661         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), THIRD_COLUMN_HEADER,
00662         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00663         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00664 
00665     jButton6.setBackground(new java.awt.Color(255, 255, 255));
00666     jButton6.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00667     jButton6.setText(UPDATE_STRING);
00668     jButton6.addActionListener(this::jButton6ActionPerformed);
00669 
00670     jButton7.setBackground(new java.awt.Color(255, 255, 255));
00671     jButton7.setFont(new java.awt.Font(SECONDARY_FONT, 0, 18)); // NOI18N
00672     jButton7.setText("Add");
00673     jButton7.addActionListener(this::jButton7ActionPerformed);
00674 
00675     javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
00676     jPanel4.setLayout(jPanel4Layout);
00677     jPanel4Layout.setHorizontalGroup(
00678         jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00679             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel4Layout.createSequentialGroup()
00680                 .addGap(0, 27, Short.MAX_VALUE)
00681                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00682                     .addGroup(jPanel4Layout.createSequentialGroup()
00683                         .addGap(24, 24, 24)
00684 
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
00685                             .addGroup(jPanel4Layout.createSequentialGroup()
00686                                 .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
00687                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00688                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
      109,
00689                                     Short.MAX_VALUE)
00690                                 .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
00691                                     javax.swing.GroupLayout.PREFERRED_SIZE)
00692                                 .addGap(8, 8, 8))
00693                             .addComponent(itemprice2)
00694                             .addComponent(itemname2, javax.swing.GroupLayout.DEFAULT_SIZE, 311,
      Short.MAX_VALUE)
00695                             .addComponent(jLabel8, javax.swing.GroupLayout.DEFAULT_SIZE,
00696                                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
00697                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00698                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00699 
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00700                             .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 586,
00701                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00702                             .addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED_SIZE, 408,
00703                                 javax.swing.GroupLayout.PREFERRED_SIZE))
00704                         .addGap(115, 115, 115))
00705                     .addGroup(jPanel4Layout.createSequentialGroup()
00706                         .addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE, 1111,
00707                             javax.swing.GroupLayout.PREFERRED_SIZE)
00708                         .addGap(18, 18, 18)))));
00709     jPanel4Layout.setVerticalGroup(
00710         jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00711             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel4Layout.createSequentialGroup()
00712                 .addContainerGap()
```

```
00713                 .addComponent(jLabel9)
00714                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 21,
      Short.MAX_VALUE)
00715
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00716                     .addComponent(jLabel8)
00717                     .addComponent(jLabel10))
00718                 .addGap(18, 18, 18)
00719                 .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00720                     .addGroup(jPanel4Layout.createSequentialGroup()
00721                         .addComponent(itemname2, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00722                             javax.swing.GroupLayout.PREFERRED_SIZE)
00723                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00724                         .addComponent(itemprice2, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00725                             javax.swing.GroupLayout.PREFERRED_SIZE)
00726                         .addGap(37, 37, 37)
00727
      .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
00728                             .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00729                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00730                             .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
00731                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00732                     .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 203,
00733                         javax.swing.GroupLayout.PREFERRED_SIZE))
00734                 .addGap(48, 48, 48)));
00735
00736     jTabbedPane1.addTab("MainCourse", jPanel4);
00737
00738     jButton3.setBackground(new java.awt.Color(255, 255, 255));
00739     jButton3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00740     jButton3.setText("Go Back");
00741     jButton3.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true));
00742     jButton3.addActionListener(this::jButton3ActionPerformed);
00743
00744     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00745     jPanel1.setLayout(jPanel1Layout);
00746     jPanel1Layout.setHorizontalGroup(
00747         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00748             .addGroup(jPanel1Layout.createSequentialGroup()
00749                 .addGap(25, 25, 25)
00750                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00751                     .addGroup(jPanel1Layout.createSequentialGroup()
00752                         .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 122,
00753                             javax.swing.GroupLayout.PREFERRED_SIZE)
00754                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
00755                             javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00756                         .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 689,
00757                             javax.swing.GroupLayout.PREFERRED_SIZE)
00758                         .addGap(218, 218, 218))
00759                     .addGroup(jPanel1Layout.createSequentialGroup()
00760                         .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
00761                             javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
00762                         .addContainerGap(29, Short.MAX_VALUE)))));
00763     jPanel1Layout.setVerticalGroup(
00764         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00765             .addGroup(jPanel1Layout.createSequentialGroup()
00766                 .addContainerGap()
00767
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
00768                     .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 60,
00769                         javax.swing.GroupLayout.PREFERRED_SIZE)
00770                     .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 37,
00771                         javax.swing.GroupLayout.PREFERRED_SIZE))
00772                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
00773                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
00774                 .addComponent(jTabbedPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 392,
00775                     javax.swing.GroupLayout.PREFERRED_SIZE)
00776                 .addGap(29, 29, 29)));
00777
00778     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00779     getContentPane().setLayout(layout);
00780     layout.setHorizontalGroup(
00781         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00782             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00783                 Short.MAX_VALUE));
00784     layout.setVerticalGroup(
00785         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00786             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
00787                 .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00788                     javax.swing.GroupLayout.PREFERRED_SIZE)
00789                 .addGap(0, 0, Short.MAX_VALUE)));
00790
00791     pack();
00792     setLocationRelativeTo(null);
```

```
00793   }// </editor-fold>//GEN-END:initComponents
00794
00805   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton2ActionPerformed
00806     String name = itemname.getText();
00807     String price = itemprice.getText();
00808
00809     new Thread(() -> {
00810       try {
00811         productService.addDrink(name, price);
00812         SwingUtilities.invokeLater(() -> {
00813           JOptionPane.showMessageDialog(this, "Drink Added Successfully.");
00814           itemname.setText("");
00815           itemprice.setText("");
00816           loadDrinks();
00817         });
00818       } catch (Exception ex) {
00819         SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog(this, "Error adding drink: " +
      ex.getMessage()));
00820       }
00821     }).start();
00822   }// GEN-LAST:event_jButton2ActionPerformed
00823
00832   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton3ActionPerformed
00833     new AdminLogin().setVisible(true);
00834     this.dispose();
00835   }// GEN-LAST:event_jButton3ActionPerformed
00836
00845   private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton5ActionPerformed
00846     String name = itemname1.getText();
00847     String price = itemprice1.getText();
00848
00849     new Thread(() -> {
00850       try {
00851         productService.addAppetizer(name, price);
00852         SwingUtilities.invokeLater(() -> {
00853           JOptionPane.showMessageDialog(this, "Appetizer Added Successfully.");
00854           itemname1.setText("");
00855           itemprice1.setText("");
00856           loadAppetizer();
00857         });
00858       } catch (Exception ex) {
00859         SwingUtilities
00860             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error adding appetizer: " +
      ex.getMessage()));
00861       }
00862     }).start();
00863   }// GEN-LAST:event_jButton5ActionPerformed
00864
00873   private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton7ActionPerformed
00874     String name = itemname2.getText();
00875     String price = itemprice2.getText();
00876
00877     new Thread(() -> {
00878       try {
00879         productService.addMainCourse(name, price);
00880         SwingUtilities.invokeLater(() -> {
00881           JOptionPane.showMessageDialog(this, "Main Course Added Successfully.");
00882           itemname2.setText("");
00883           itemprice2.setText("");
00884           loadmainCourse();
00885         });
00886       } catch (Exception ex) {
00887         SwingUtilities
00888             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error adding main course: " +
      ex.getMessage()));
00889       }
00890     }).start();
00891   }// GEN-LAST:event_jButton7ActionPerformed
00892
00901   private void jTable1KeyPressed(java.awt.event.KeyEvent evt) {// GEN-FIRST:event_jTable1KeyPressed
00902     // No action needed for key press in this version.
00903   }// GEN-LAST:event_jTable1KeyPressed
00904
00915   private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {//
      GEN-FIRST:event_jTable1MouseClicked
00916     int row = jTable1.getSelectedRow();
00917     if (row == -1)
00918       return;
00919
00920     String idStr = jTable1.getValueAt(row, 0).toString();
00921     selectedDrinkID = Long.valueOf(idStr);
00922     LOGGER.info("Selected Drink ID: {}", selectedDrinkID);
00923
```

```
00924     new Thread(() -> {
00925       try {
00926         Drink drink = productService.getDrinkById(selectedDrinkID);
00927         SwingUtilities.invokeLater(() -> {
00928           itemname.setText(drink.getItemDrinks());
00929           itemprice.setText(String.valueOf(drink.getDrinksPrice()));
00930         });
00931       } catch (Exception ex) {
00932         SwingUtilities
00933             .invokeLater(() -> JOptionPane.showMessageDialog(null, "Error loading drink details: " +
      ex.getMessage()));
00934       }
00935     }).start();
00936   }// GEN-LAST:event_jTable1MouseClicked
00937
00946   private void jTable2MouseClicked(java.awt.event.MouseEvent evt) {//
      GEN-FIRST:event_jTable2MouseClicked
00947     int row = jTable2.getSelectedRow();
00948     if (row == -1)
00949       return;
00950
00951     String idStr = jTable2.getValueAt(row, 0).toString();
00952     selectedAppetizerID = Long.valueOf(idStr);
00953     LOGGER.info("Selected Appetizer ID: {}", selectedAppetizerID);
00954
00955     new Thread(() -> {
00956       try {
00957         Appetizer appetizer = productService.getAppetizerById(selectedAppetizerID);
00958         SwingUtilities.invokeLater(() -> {
00959           itemname1.setText(appetizer.getItemAppetizers());
00960           itemprice1.setText(String.valueOf(appetizer.getAppetizersPrice()));
00961         });
00962       } catch (Exception ex) {
00963         SwingUtilities.invokeLater(
00964             () -> JOptionPane.showMessageDialog(null, "Error loading appetizer details: " +
      ex.getMessage()));
00965       }
00966     }).start();
00967   }// GEN-LAST:event_jTable2MouseClicked
00968
00977   private void jTable3MouseClicked(java.awt.event.MouseEvent evt) {//
      GEN-FIRST:event_jTable3MouseClicked
00978     int row = jTable3.getSelectedRow();
00979     if (row == -1)
00980       return;
00981
00982     String idStr = jTable3.getValueAt(row, 0).toString();
00983     selectedMainCourseID = Long.valueOf(idStr);
00984     LOGGER.info("Selected Main Course ID: {}", selectedMainCourseID);
00985
00986     new Thread(() -> {
00987       try {
00988         MainCourse mainCourse = productService.getMainCourseById(selectedMainCourseID);
00989         SwingUtilities.invokeLater(() -> {
00990           itemname2.setText(mainCourse.getItemFood());
00991           itemprice2.setText(String.valueOf(mainCourse.getFoodPrice()));
00992         });
00993       } catch (Exception ex) {
00994         SwingUtilities.invokeLater(
00995             () -> JOptionPane.showMessageDialog(null, "Error loading main course details: " +
      ex.getMessage()));
00996       }
00997     }).start();
00998   }// GEN-LAST:event_jTable3MouseClicked
00999
01008   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton1ActionPerformed
01009     String name = itemname.getText();
01010     String price = itemprice.getText();
01011
01012     new Thread(() -> {
01013       try {
01014         productService.updateDrink(selectedDrinkID, name, price);
01015         SwingUtilities.invokeLater(() -> {
01016           JOptionPane.showMessageDialog(this, "Drink updated successfully.");
01017           itemname.setText("");
01018           itemprice.setText("");
01019           selectedDrinkID = null;
01020           loadDrinks();
01021         });
01022       } catch (Exception ex) {
01023         SwingUtilities
01024             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error updating drink: " +
      ex.getMessage()));
01025       }
01026     }).start();
01027   }// GEN-LAST:event_jButton1ActionPerformed
```

```
01028
01037   private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton4ActionPerformed
01038     String name = itemname1.getText();
01039     String price = itemprice1.getText();
01040
01041     new Thread(() -> {
01042       try {
01043         productService.updateAppetizer(selectedAppetizerID, name, price);
01044         SwingUtilities.invokeLater(() -> {
01045           JOptionPane.showMessageDialog(this, "Appetizer updated successfully.");
01046           itemname1.setText("");
01047           itemprice1.setText("");
01048           selectedAppetizerID = null;
01049           loadAppetizer();
01050         });
01051       } catch (Exception ex) {
01052         SwingUtilities
01053             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error updating appetizer: " +
      ex.getMessage()));
01054       }
01055     }).start();
01056   }// GEN-LAST:event_jButton4ActionPerformed
01057
01066   private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton6ActionPerformed
01067     String name = itemname2.getText();
01068     String price = itemprice2.getText();
01069
01070     new Thread(() -> {
01071       try {
01072         productService.updateMainCourse(selectedMainCourseID, name, price);
01073         SwingUtilities.invokeLater(() -> {
01074           JOptionPane.showMessageDialog(this, "Main course updated successfully.");
01075           itemname2.setText("");
01076           itemprice2.setText("");
01077           selectedMainCourseID = null;
01078           loadmainCourse();
01079         });
01080       } catch (Exception ex) {
01081         SwingUtilities
01082             .invokeLater(() -> JOptionPane.showMessageDialog(this, "Error updating main course: " +
      ex.getMessage()));
01083       }
01084     }).start();
01085   }// GEN-LAST:event_jButton6ActionPerformed
01086
01096   public static void main(String[] args) {
01097     try {
01098       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
01099         if ("Nimbus".equals(info.getName())) {
01100           javax.swing.UIManager.setLookAndFeel(info.getClassName());
01101           break;
01102         }
01103       }
01104     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException |
      javax.swing.UnsupportedLookAndFeelException ex) {
01105
      java.util.logging.Logger.getLogger(AdminProducts.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
01106     }
01107     // </editor-fold>
01108
01109     /* Create and display the form */
01110     java.awt.EventQueue.invokeLater(() -> new AdminProducts().setVisible(true));
01111   }
01112
01113   // Variables declaration - do not modify//GEN-BEGIN:variables
01114   // Sonarqube rule java:S1450 must be ignored here as these variables are
01115   // auto-generated by the Form Editor and need to remain as instance variables.
01117   javax.swing.JTextField itemname;
01119   javax.swing.JTextField itemname1;
01121   javax.swing.JTextField itemname2;
01123   javax.swing.JTextField itemprice;
01125   javax.swing.JTextField itemprice1;
01127   javax.swing.JTextField itemprice2;
01129   javax.swing.JButton jButton1;
01131   javax.swing.JButton jButton2;
01133   javax.swing.JButton jButton3;
01135   javax.swing.JButton jButton4;
01137   javax.swing.JButton jButton5;
01139   javax.swing.JButton jButton6;
01141   javax.swing.JButton jButton7;
01143   javax.swing.JLabel jLabel1;
01145   javax.swing.JLabel jLabel10;
01147   javax.swing.JLabel jLabel2;
```

```
01149   javax.swing.JLabel jLabel3;
01151   javax.swing.JLabel jLabel4;
01153   javax.swing.JLabel jLabel5;
01155   javax.swing.JLabel jLabel6;
01157   javax.swing.JLabel jLabel7;
01159   javax.swing.JLabel jLabel8;
01161   javax.swing.JLabel jLabel9;
01163   javax.swing.JPanel jPanel1;
01165   javax.swing.JPanel jPanel2;
01167   javax.swing.JPanel jPanel3;
01169   javax.swing.JPanel jPanel4;
01171   javax.swing.JScrollPane jScrollPane1;
01173   javax.swing.JScrollPane jScrollPane2;
01175   javax.swing.JScrollPane jScrollPane3;
01177   javax.swing.JTabbedPane jTabbedPane1;
01179   javax.swing.JTable jTable1;
01181   javax.swing.JTable jTable2;
01183   javax.swing.JTable jTable3;
01184   // End of variables declaration//GEN-END:variables
01185 }
```

10.50 ApplicationLauncher.java File Reference

Classes

- class es.ull.esit.app.ApplicationLauncher

  Auxiliary entry point used to start the application's UI.

Packages

- package es.ull.esit.app

10.51 ApplicationLauncher.java

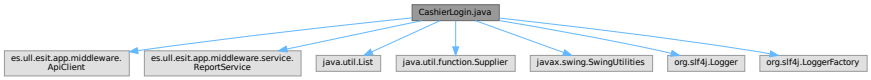
Go to the documentation of this file.

```
00001 package es.ull.esit.app;
00002
00010 public class ApplicationLauncher {
00011
00019   public static void main(String[] args) {
00020     // Use a factory so tests can inject a mock Login instance and avoid
00021     // creating real GUI components (which can fail in headless CI).
00022     java.awt.EventQueue.invokeLater(() -> getLoginFactory().get().setVisible(true));
00023   }
00024
00025   // ----------------------------------------------------------------------
00026   // Test seam: a supplier that creates the Login instance. By default it
00027   // constructs a real Login, but tests can replace it with a supplier that
00028   // returns a mock to avoid UI creation.
00029   // ----------------------------------------------------------------------
00030   private static java.util.function.Supplier<Login> loginFactory = Login::new;
00031
00036   static void setLoginFactory(java.util.function.Supplier<Login> factory) {
00037     loginFactory = factory;
00038   }
00039
00040   static java.util.function.Supplier<Login> getLoginFactory() {
00041     return loginFactory;
00042   }
00043 }
```

10.52 CashierLogin.java File Reference

```
import es.ull.esit.app.middleware.ApiClient;
import es.ull.esit.app.middleware.service.ReportService;
import java.util.List;
import java.util.function.Supplier;
import javax.swing.SwingUtilities;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

Include dependency graph for CashierLogin.java:



Classes

- class es.ull.esit.app.CashierLogin

  Login window for authenticating cashiers.

Packages

- package es.ull.esit.app

10.53 CashierLogin.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app;
00002
00003 import es.ull.esit.app.middleware.ApiClient;
00004 import es.ull.esit.app.middleware.service.ReportService;
00005 import java.util.List;
00006 import java.util.function.Supplier;
00007 import javax.swing.SwingUtilities;
00008 import org.slf4j.Logger;
00009 import org.slf4j.LoggerFactory;
00010
00011
00027 public class CashierLogin extends javax.swing.JFrame {
00028
00030   private final transient ReportService reportService;
00031
00032   private static final String PRIMARY_FONT = "Yu Gothic UI";
00033
00035   private static final Logger LOGGER = LoggerFactory.getLogger(CashierLogin.class);
00036
00044   public CashierLogin() {
00045     this((String) null);
00046   }
00047
00058   public CashierLogin(String name) {
00059     initComponents();
00060
00061     ApiClient client = new ApiClient("http://localhost:8080");
00062     this.reportService = new ReportService(client);
00063
00064     updateWelcomeMessage(name);
00065   }
00066
00075   CashierLogin(ReportService reportService, String name, boolean startBackground) {
00076     initComponents();
```

```
00077     this.reportService = reportService;
00078     updateWelcomeMessage(name);
00079     // do not start background threads in tests; if requested, callers may
00080     // manually invoke the async handlers or startBackground could be used
00081     // in future to mimic the default constructor behaviour.
00082     if (startBackground) {
00083       // no-op for now; left intentionally minimal to avoid side-effects in tests
00084     }
00085   }
00086
00087   /*
00088    * Suppliers for windows created by the action handlers. These allow tests
00089    * to inject no-op or stub frames so the asynchronous handlers don't open
00090    * real GUI windows during unit tests.
00091    */
00092   Supplier<? extends javax.swing.JFrame> aboutUsSupplier = () -> new AboutUs();
00093   Supplier<? extends javax.swing.JFrame> orderSupplier = () -> new Order();
00094   Supplier<? extends javax.swing.JFrame> loginSupplier = () -> new Login();
00095
00097   private static Supplier<? extends javax.swing.JFrame> cashierSupplier = () -> new CashierLogin();
00098
00099   static void setCashierSupplier(Supplier<? extends javax.swing.JFrame> s) {
00100     cashierSupplier = s;
00101   }
00102
00103   static Supplier<? extends javax.swing.JFrame> getCashierSupplier() {
00104     return cashierSupplier;
00105   }
00106
00108   void setAboutUsSupplier(Supplier<? extends javax.swing.JFrame> s) {
00109     this.aboutUsSupplier = s;
00110   }
00111
00112   void setOrderSupplier(Supplier<? extends javax.swing.JFrame> s) {
00113     this.orderSupplier = s;
00114   }
00115
00116   void setLoginSupplier(Supplier<? extends javax.swing.JFrame> s) {
00117     this.loginSupplier = s;
00118   }
00119
00125   void aboutUsActionRun() {
00126     try {
00127       List<es.ull.esit.app.middleware.model.Cashier> cashiers = reportService.getCashierInfo();
00128       LOGGER.info("Found {} cashiers in DB.", cashiers == null ? 0 : cashiers.size());
00129     } catch (Exception ex) {
00130       LOGGER.warn("Warning: Could not fetch cashier stats: {}", ex.getMessage());
00131     }
00132     // Use supplier so tests can inject a no-op frame
00133     javax.swing.JFrame f = aboutUsSupplier.get();
00134     f.setVisible(true);
00135     dispose();
00136   }
00137
00142   void menuActionRun() {
00143     try {
00144       String status = reportService.checkMenuStatus();
00145       LOGGER.info("Menu status: {}", status);
00146     } catch (Exception ex) {
00147       LOGGER.error("Error checking menu status: {}", ex.getMessage());
00148     }
00149     javax.swing.JFrame f = orderSupplier.get();
00150     f.setVisible(true);
00151     dispose();
00152   }
00153
00158   void logoutActionRun() {
00159     javax.swing.JFrame f = loginSupplier.get();
00160     f.setVisible(true);
00161     dispose();
00162   }
00163
00170   void testOnlyCoverageHelper() {
00171     // Initialize 'a' from a runtime-varying value so the condition below
00172     // does not always evaluate to true (avoids a constant-condition
00173     // reliability issue reported by Sonar). Using currentTimeMillis() % 2
00174     // is lightweight and deterministic enough for test coverage purposes
00175     // while preventing a constant-true branch.
00176     int a = (int) (System.currentTimeMillis() % 2);
00177     if (a == 0) {
00178       a = 1;
00179     } else {
00180       a = -1;
00181     }
00182     switch (a) {
00183       case 1:
00184         a++;
```

```
00185         break;
00186       default:
00187         a--;
00188     }
00189     String tmp = "ok" + a;
00190     LOGGER.debug("coverage helper: {}", tmp);
00191   }
00192
00201   private void updateWelcomeMessage(String name) {
00202     if (name == null || name.trim().isEmpty()) {
00203       welcomeTxt.setText("Welcome Cashier");
00204     } else {
00205       welcomeTxt.setText("Welcome " + name);
00206     }
00207   }
00208
00225   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
00226     new Thread(() -> {
00227       try {
00228         List<es.ull.esit.app.middleware.model.Cashier> cashiers = reportService.getCashierInfo();
00229         LOGGER.info("Found {} cashiers in DB.", cashiers.size());
00230
00231       } catch (Exception ex) {
00232         LOGGER.warn("Warning: Could not fetch cashier stats: {}", ex.getMessage());
00233       }
00234       SwingUtilities.invokeLater(() -> {
00235         aboutUsSupplier.get().setVisible(true);
00236         dispose();
00237       });
00238     }).start();
00239   }
00240
00245   int aboutUsActionSync() {
00246     try {
00247       List<es.ull.esit.app.middleware.model.Cashier> cashiers = reportService.getCashierInfo();
00248       return cashiers == null ? 0 : cashiers.size();
00249     } catch (Exception ex) {
00250       LOGGER.warn("Warning: Could not fetch cashier stats: {}", ex.getMessage());
00251       return -1;
00252     }
00253   }
00254
00272   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
00273     new Thread(() -> {
00274       try {
00275         String status = reportService.checkMenuStatus();
00276         LOGGER.info("Menu status: {}", status);
00277
00278       } catch (Exception ex) {
00279         LOGGER.error("Error checking menu status: {}", ex.getMessage());
00280       }
00281
00282       SwingUtilities.invokeLater(() -> {
00283         orderSupplier.get().setVisible(true);
00284         dispose();
00285       });
00286     }).start();
00287   }
00288
00293   String menuActionSync() {
00294     try {
00295       return reportService.checkMenuStatus();
00296     } catch (Exception ex) {
00297       LOGGER.error("Error checking menu status: {}", ex.getMessage());
00298       return null;
00299     }
00300   }
00301
00315   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
00316     loginSupplier.get().setVisible(true);
00317     dispose();
00318   }
00319
00324   boolean logoutActionSync() {
00325     // In production this would open the Login window and dispose(). For tests
00326     // we simply indicate the action was executed.
00327     return true;
00328   }
00329
00331   String getWelcomeText() {
00332     return welcomeTxt.getText();
00333   }
00334
00345   public static void main(String[] args) {
00346     try {
00347       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
```

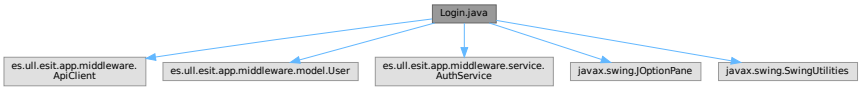
```
00348         if ("Nimbus".equals(info.getName())) {
00349           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00350           break;
00351         }
00352       }
00353     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00354         | javax.swing.UnsupportedLookAndFeelException ex) {
00355 
      java.util.logging.Logger.getLogger(CashierLogin.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00356     }
00357 
00358     java.awt.EventQueue.invokeLater(() -> getCashierSupplier().get().setVisible(true));
00359   }
00360 
00369   @SuppressWarnings("unchecked")
00370   // <editor-fold defaultstate="collapsed" desc="Generated
00371   // Code">//GEN-BEGIN:initComponents
00372   private void initComponents() {
00373 
00374     jPanel1 = new javax.swing.JPanel();
00375     welcomeTxt = new javax.swing.JLabel();
00376     jButton1 = new javax.swing.JButton();
00377     jButton2 = new javax.swing.JButton();
00378     jLabel1 = new javax.swing.JLabel();
00379     jButton3 = new javax.swing.JButton();
00380 
00381     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00382     setTitle("Cashier");
00383     setResizable(false);
00384 
00385     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00386 
00387     welcomeTxt.setFont(new java.awt.Font(PRIMARY_FONT, 1, 24)); // NOI18N
00388     welcomeTxt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00389     welcomeTxt.setText("Welcome Cashier");
00390 
00391     jButton1.setBackground(new java.awt.Color(153, 153, 153));
00392     jButton1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00393     jButton1.setText("About us");
00394     jButton1.addActionListener(this::jButton1ActionPerformed);
00395 
00396     jButton2.setBackground(new java.awt.Color(153, 153, 153));
00397     jButton2.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00398     jButton2.setText("Menu");
00399     jButton2.addActionListener(this::jButton2ActionPerformed);
00400 
00401     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
      NOI18N
00402 
00403     jButton3.setBackground(new java.awt.Color(255, 255, 255));
00404     jButton3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00405     jButton3.setText("LogOut");
00406     jButton3.addActionListener(this::jButton3ActionPerformed);
00407 
00408     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00409     jPanel1.setLayout(jPanel1Layout);
00410     jPanel1Layout.setHorizontalGroup(
00411         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00412             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00413                 .addContainerGap(109, Short.MAX_VALUE)
00414                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00415                     .addComponent(welcomeTxt, javax.swing.GroupLayout.PREFERRED_SIZE, 299,
00416                         javax.swing.GroupLayout.PREFERRED_SIZE)
00417 
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00418                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00419                             .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00420                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00421                             .addGap(98, 98, 98))
00422                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00423                             .addComponent(jLabel1)
00424                             .addGap(183, 183, 183))
00425                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00426                             .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00427                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00428                             .addGap(190, 190, 190)))))
00429             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00430                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00431                     .addContainerGap(108, Short.MAX_VALUE)
00432                     .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 320,
00433                         javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
00434                     .addGap(99, 99, 99))));
00435     jPanel1Layout.setVerticalGroup(
00436         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00437             .addGroup(jPanel1Layout.createSequentialGroup()
00438                 .addGap(45, 45, 45)
00439                 .addComponent(welcomeTxt, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
00440                     javax.swing.GroupLayout.PREFERRED_SIZE)
00441                 .addGap(18, 18, 18)
00442                 .addComponent(jLabel1)
00443                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 99,
      Short.MAX_VALUE)
00444                 .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00445                     javax.swing.GroupLayout.PREFERRED_SIZE)
00446                 .addGap(57, 57, 57)
00447                 .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00448                     javax.swing.GroupLayout.PREFERRED_SIZE)
00449                 .addGap(68, 68, 68))
00450             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00451                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      jPanel1Layout.createSequentialGroup()
00452                     .addContainerGap(290, Short.MAX_VALUE)
00453                     .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00454                         javax.swing.GroupLayout.PREFERRED_SIZE)
00455                     .addGap(242, 242, 242))));
00456
00457     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00458     getContentPane().setLayout(layout);
00459     layout.setHorizontalGroup(
00460         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00461             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00462                 Short.MAX_VALUE));
00463     layout.setVerticalGroup(
00464         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00465             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00466                 Short.MAX_VALUE));
00467
00468     pack();
00469     setLocationRelativeTo(null);
00470   }// </editor-fold>//GEN-END:initComponents
00471
00472   // Variables declaration - do not modify//GEN-BEGIN:variables
00473   // Sonarqube rule java:S1450 must be ignored here as these variables are
00474   // auto-generated by the Form Editor and need to remain as instance variables.
00476   private javax.swing.JButton jButton1;
00478   private javax.swing.JButton jButton2;
00480   private javax.swing.JButton jButton3;
00482   private javax.swing.JLabel jLabel1;
00484   private javax.swing.JPanel jPanel1;
00486   private javax.swing.JLabel welcomeTxt;
00487   // End of variables declaration//GEN-END:variables
00488 }
```

10.54 Login.java File Reference

```
import es.ull.esit.app.middleware.ApiClient;
import es.ull.esit.app.middleware.model.User;
import es.ull.esit.app.middleware.service.AuthService;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;
```

Include dependency graph for Login.java:



Classes

- class es.ull.esit.app.Login

Login window for the Restaurant System.

- interface es.ull.esit.app.Login.MessageDialog

Packages

- package es.ull.esit.app

10.55 Login.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app;
00002
00003 import es.ull.esit.app.middleware.ApiClient;
00004 import es.ull.esit.app.middleware.model.User;
00005 import es.ull.esit.app.middleware.service.AuthService;
00006 import javax.swing.JOptionPane;
00007 import javax.swing.SwingUtilities;
00008
00023 public class Login extends javax.swing.JFrame {
00024
00026   private final transient AuthService authService;
00027
00028   // --- Test seams: allow tests to inject frames and dialog handlers to avoid
00029   // creating real UI windows or showing JOptionPane dialogs during unit tests.
00030   java.util.function.Supplier<? extends javax.swing.JFrame> adminSupplier = () -> new AdminLogin();
00031   java.util.function.Function<String, ? extends javax.swing.JFrame> cashierFactory =
      CashierLogin::new;
00032
00033   interface MessageDialog {
00034     void showMessage(java.awt.Component parent, Object message);
00035
00036     void showMessage(java.awt.Component parent, Object message, String title, int messageType);
00037   }
00038
00039   MessageDialog messageDialog = new MessageDialog() {
00040     @Override
00041     public void showMessage(java.awt.Component parent, Object message) {
00042       JOptionPane.showMessageDialog(parent, message);
00043     }
00044
00045     @Override
00046     public void showMessage(java.awt.Component parent, Object message, String title, int messageType)
      {
00047       JOptionPane.showMessageDialog(parent, message, title, messageType);
00048     }
00049   };
00050
00051   void setAdminSupplier(java.util.function.Supplier<? extends javax.swing.JFrame> s) {
00052     this.adminSupplier = s;
00053   }
00054
00055   void setCashierFactory(java.util.function.Function<String, ? extends javax.swing.JFrame> f) {
00056     this.cashierFactory = f;
00057   }
00058
00059   void setMessageDialog(MessageDialog md) {
00060     this.messageDialog = md;
00061   }
00062
00064   private static final String PRIMARY_FONT = "Yu Gothic UI";
00065
00067   private static final String LOGIN_STRING = "Log in";
00068
00078   public Login() {
00079     initComponents();
00080     ApiClient client = new ApiClient("http://localhost:8080");
00081     this.authService = new AuthService(client);
00082   }
00083
00090   @SuppressWarnings("unchecked")
00091   // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
00092   private void initComponents() {
00093
00094     jPanel1 = new javax.swing.JPanel();
00095     jLabel1 = new javax.swing.JLabel();
00096     jLabel3 = new javax.swing.JLabel();
```

```
00097     usernametxt = new javax.swing.JTextField();
00098     usertypecmbo = new javax.swing.JComboBox<>();
00099     jButton1 = new javax.swing.JButton();
00100     jPasswordField1 = new javax.swing.JPasswordField();
00101 
00102     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00103     setTitle("Login");
00104     setResizable(false);
00105 
00106     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00107 
00108     jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
      NOI18N
00109 
00110     jLabel3.setFont(new java.awt.Font(PRIMARY_FONT, 1, 24)); // NOI18N
00111     jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00112     jLabel3.setText("User Login ");
00113 
00114     usernametxt.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00115     usernametxt.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00116     usernametxt.setBorder(javax.swing.BorderFactory.createTitledBorder(
00117         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Username",
00118         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00119         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00120 
00121     usertypecmbo.setEditable(true);
00122     usertypecmbo.setFont(new java.awt.Font(PRIMARY_FONT, 0, 18)); // NOI18N
00123     usertypecmbo.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "admin", "cashier"
      }));
00124     usertypecmbo.addActionListener(this::usertypecmboActionPerformed);
00125 
00126     jButton1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 18)); // NOI18N
00127     jButton1.setText(LOGIN_STRING);
00128     jButton1.addActionListener(this::jButton1ActionPerformed);
00129 
00130     jPasswordField1.setFont(new java.awt.Font("Thonburi", 0, 18)); // NOI18N
00131     jPasswordField1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00132     jPasswordField1.setBorder(javax.swing.BorderFactory.createTitledBorder(
00133         javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)), "Password",
00134         javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP,
00135         new java.awt.Font("Lucida Grande", 0, 18))); // NOI18N
00136 
00137     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00138     jPanel1.setLayout(jPanel1Layout);
00139     jPanel1Layout.setHorizontalGroup(
00140         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00141             .addGroup(jPanel1Layout.createSequentialGroup()
00142                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00143                     .addGroup(jPanel1Layout.createSequentialGroup()
00144                         .addGap(132, 132, 132)
00145                         .addComponent(jLabel1))
00146                     .addGroup(jPanel1Layout.createSequentialGroup()
00147                         .addGap(61, 61, 61)
00148                         .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 309,
00149                             javax.swing.GroupLayout.PREFERRED_SIZE))
00150                     .addGroup(jPanel1Layout.createSequentialGroup()
00151                         .addGap(74, 74, 74)
00152 
      .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00153                             .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE,
      311,
00154                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00155                             .addComponent(usernametxt, javax.swing.GroupLayout.PREFERRED_SIZE, 311,
00156                                 javax.swing.GroupLayout.PREFERRED_SIZE)
00157                             .addComponent(usertypecmbo, javax.swing.GroupLayout.PREFERRED_SIZE, 311,
00158                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
00159                     .addGroup(jPanel1Layout.createSequentialGroup()
00160                         .addGap(146, 146, 146)
00161                         .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
00162                             javax.swing.GroupLayout.PREFERRED_SIZE)))
00163                 .addContainerGap(86, Short.MAX_VALUE)));
00164     jPanel1Layout.setVerticalGroup(
00165         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00166             .addGroup(jPanel1Layout.createSequentialGroup()
00167                 .addGap(39, 39, 39)
00168                 .addComponent(jLabel1)
00169                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00170                 .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
00171                     javax.swing.GroupLayout.PREFERRED_SIZE)
00172                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00173                 .addComponent(usernametxt, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00174                     javax.swing.GroupLayout.PREFERRED_SIZE)
00175                 .addGap(18, 18, 18)
00176                 .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
00177                     javax.swing.GroupLayout.PREFERRED_SIZE)
00178                 .addGap(18, 18, 18)
00179                 .addComponent(usertypecmbo, javax.swing.GroupLayout.PREFERRED_SIZE, 46,
```

```
00180                   javax.swing.GroupLayout.PREFERRED_SIZE)
00181               .addGap(18, 18, 18)
00182               .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 49,
00183                   javax.swing.GroupLayout.PREFERRED_SIZE)
00184               .addContainerGap(39, Short.MAX_VALUE)));
00185
00186     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00187     getContentPane().setLayout(layout);
00188     layout.setHorizontalGroup(
00189         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00190             .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00191                 javax.swing.GroupLayout.PREFERRED_SIZE));
00192     layout.setVerticalGroup(
00193         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00194             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00195                 Short.MAX_VALUE));
00196
00197     pack();
00198     setLocationRelativeTo(null);
00199   }// </editor-fold>//GEN-END:initComponents
00200
00214   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_jButton1ActionPerformed
00215     // Get username and password from input fields.
00216     String usernameInput = usernametxt.getText();
00217     String passwordInput = new String(jPasswordField1.getPassword());
00218
00219     jButton1.setEnabled(false);
00220     jButton1.setText("Loading...");
00221
00222     new Thread(() -> {
00223       try {
00224         // Authenticate user through AuthService.
00225         User loggedInUser = authService.authenticate(usernameInput, passwordInput);
00226
00227         SwingUtilities.invokeLater(() -> {
00228
00229           // Open the corresponding window based on user role. Use the
00230           // injectable factories so tests can substitute frames.
00231           if ("ADMIN".equalsIgnoreCase(loggedInUser.getRole())) {
00232             adminSupplier.get().setVisible(true);
00233           } else if ("CASHIER".equalsIgnoreCase(loggedInUser.getRole())) {
00234             cashierFactory.apply(loggedInUser.getUsername()).setVisible(true);
00235           } else {
00236             messageDialog.showMessage(this, "Unknown Role: " + loggedInUser.getRole());
00237             jButton1.setEnabled(true);
00238             jButton1.setText(LOGIN_STRING);
00239             return;
00240           }
00241
00242           this.dispose();
00243         });
00244
00245       } catch (Exception e) {
00246         SwingUtilities.invokeLater(() -> {
00247           messageDialog.showMessage(this, "Login failed: " + e.getMessage(), "Login Error",
      JOptionPane.ERROR_MESSAGE);
00248
00249           jButton1.setEnabled(true);
00250           jButton1.setText(LOGIN_STRING);
00251         });
00252       }
00253     }).start();
00254   }// GEN-LAST:event_jButton1ActionPerformed
00255
00264   private void usertypecmboActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_usertypecmboActionPerformed
00265     // No action needed here.
00266   }// GEN-LAST:event_usertypecmboActionPerformed
00267
00273   public static void main(String[] args) {
00274     try {
00275       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00276         if ("Nimbus".equals(info.getName())) {
00277           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00278           break;
00279         }
00280       }
00281     } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
00282         | javax.swing.UnsupportedLookAndFeelException ex) {
00283       java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00284     }
00285
```

```
00286     java.awt.EventQueue.invokeLater(() -> getLoginFactory().get().setVisible(true));
00287   }
00288
00289   // Test seam for main: allow tests to replace the constructor of Login
00290   private static java.util.function.Supplier<Login> loginFactory = Login::new;
00291
00292   static void setLoginFactory(java.util.function.Supplier<Login> f) {
00293     loginFactory = f;
00294   }
00295
00296   static java.util.function.Supplier<Login> getLoginFactory() {
00297     return loginFactory;
00298   }
00299
00300   // Variables declaration - do not modify//GEN-BEGIN:variables
00301   // Sonarqube rule java:S1450 must be ignored here as these variables are
00302   // auto-generated by the Form Editor and need to remain as instance variables.
00304   private javax.swing.JButton jButton1;
00306   private javax.swing.JLabel jLabel1;
00308   private javax.swing.JLabel jLabel3;
00310   private javax.swing.JPanel jPanel1;
00312   private javax.swing.JPasswordField jPasswordField1;
00314   private javax.swing.JTextField usernametxt;
00319   private javax.swing.JComboBox<String> usertypecmbo;
00320   // End of variables declaration//GEN-END:variables
00321 }
```

10.56 ApiClient.java File Reference

```
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.Duration;
import java.util.List;
import java.util.Map;
import es.ull.esit.app.middleware.model.Appetizer;
import es.ull.esit.app.middleware.model.Cashier;
import es.ull.esit.app.middleware.model.Drink;
import es.ull.esit.app.middleware.model.MainCourse;
import es.ull.esit.app.middleware.model.User;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

Include dependency graph for ApiClient.java:

Classes

- class es.ull.esit.app.middleware.ApiClient

  API client for interacting with the backend REST API.
- interface es.ull.esit.app.middleware.ApiClient.IOCallable< T >

  Functional interface for lambdas that may throw InterruptedException or IOException.
- interface es.ull.esit.app.middleware.ApiClient.ResponseHandler< R >

  Functional interface for processing HTTP responses and possibly throwing IOException from JSON parsing.

Packages

- package es.ull.esit.app.middleware

10.57 ApiClient.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware;
00002
00003 import com.fasterxml.jackson.core.type.TypeReference;
00004 import com.fasterxml.jackson.databind.ObjectMapper;
00005
00006 import java.io.IOException;
00007 import java.net.URI;
00008 import java.net.http.HttpClient;
00009 import java.net.http.HttpRequest;
00010 import java.net.http.HttpResponse;
00011 import java.time.Duration;
00012 import java.util.List;
00013 import java.util.Map;
00014
00015 import es.ull.esit.app.middleware.model.Appetizer;
00016 import es.ull.esit.app.middleware.model.Cashier;
00017 import es.ull.esit.app.middleware.model.Drink;
00018 import es.ull.esit.app.middleware.model.MainCourse;
00019 import es.ull.esit.app.middleware.model.User;
00020
00021 import org.slf4j.Logger;
00022 import org.slf4j.LoggerFactory;
00023
00031 public class ApiClient {
00032
00034   private static final String APPLICATION_JSON = "application/json";
00035
00037   private static final Logger LOGGER = LoggerFactory.getLogger(ApiClient.class);
00038
00040   private static final String CONTENT_TYPE = "Content-Type";
00041
00043   private static final String API_APPETIZERS = "/api/appetizers/";
00044
00046   private static final String API_DRINKS = "/api/drinks/";
00047
00049   private static final String API_MAINCOURSES = "/api/maincourses/";
00050
00051   private static final String API_LOGIN = "/api/login";
00052
00054   private final HttpClient http;
00055
00057   private final String baseUrl;
00058
00063   private final ObjectMapper mapper;
00064
00074   public ApiClient(String baseUrl) {
00075     this.baseUrl = baseUrl.endsWith("/") ? baseUrl.substring(0, baseUrl.length() - 1) : baseUrl;
00076     this.http = HttpClient.newBuilder()
00077         .connectTimeout(Duration.ofSeconds(5))
00078         .build();
00079     this.mapper = new ObjectMapper();
00080   }
00081
00086   ApiClient(String baseUrl, HttpClient http, ObjectMapper mapper) {
00087     this.baseUrl = baseUrl.endsWith("/") ? baseUrl.substring(0, baseUrl.length() - 1) : baseUrl;
00088     this.http = http;
00089     this.mapper = mapper == null ? new ObjectMapper() : mapper;
00090   }
00091
00095   @FunctionalInterface
00096   private interface IOCallable<T> {
00097     T call() throws InterruptedException, IOException;
00098   }
00099
00109   private <T> T execute(String action, IOCallable<T> callable) {
00110     try {
00111       return callable.call();
00112     } catch (InterruptedException e) {
00113       Thread.currentThread().interrupt();
00114       throw new ApiClientException("Thread interrupted during " + action, e);
00115     } catch (IOException e) {
00116       throw new ApiClientException("I/O error during " + action, e);
00117     }
```

```
00118   }
00119
00124   @FunctionalInterface
00125   private interface ResponseHandler<R> {
00126     R handle(int status, String body) throws IOException;
00127   }
00128
00134   private <R> R sendAndHandle(String action, HttpRequest request, ResponseHandler<R> handler)
00135       throws InterruptedException, IOException {
00136     HttpResponse<String> res = http.send(request, HttpResponse.BodyHandlers.ofString());
00137     int status = res.statusCode();
00138     String body = res.body();
00139
00140     LOGGER.info("{} -> HTTP {}", action, status);
00141     LOGGER.info("Response body: '{}'", body);
00142
00143     try {
00144       return handler.handle(status, body);
00145     } catch (IOException e) {
00146       // Wrap parsing IO exceptions with more context (status + body) so the UI
00147       // receives a helpful message instead of a generic one.
00148       throw new ApiClientException("I/O error during " + action + " status: " + status + " body: " +
      body, e);
00149     }
00150   }
00151
00152   // -------------------------------------------------------------------------
00153   // Helpers genéricos
00154   // -------------------------------------------------------------------------
00155
00169   private <T> T get(String path, Class<T> responseType) {
00170     return execute("GET " + path, () -> sendAndHandle("GET " + path,
00171         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).GET().header("Accept",
      APPLICATION_JSON).build(),
00172         (status, body) -> {
00173           if (status == 204 || body == null || body.trim().isEmpty()) {
00174             return null;
00175           }
00176           if (status < 200 || status >= 300) {
00177             throw new ApiClientException("GET " + path + " failed with HTTP " + status + " body: " +
      body);
00178           }
00179           return mapper.readValue(body, responseType);
00180         }));
00181   }
00182
00196   private <T> List<T> getList(String path, TypeReference<List<T>> typeRef) {
00197     return execute("GET " + path, () -> sendAndHandle("GET " + path,
00198         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).GET().header("Accept",
      APPLICATION_JSON).build(),
00199         (status, body) -> {
00200           if (status == 204 || body == null || body.trim().isEmpty()) {
00201             return java.util.Collections.emptyList();
00202           }
00203           if (status < 200 || status >= 300) {
00204             throw new ApiClientException("GET " + path + " failed with HTTP " + status + " body: " +
      body);
00205           }
00206           return mapper.readValue(body, typeRef);
00207         }));
00208   }
00209
00225   private <T, R> R post(String path, T body, Class<R> responseType) {
00226     return execute("POST " + path, () -> {
00227       String json = mapper.writeValueAsString(body);
00228       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00229           .POST(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
      APPLICATION_JSON).build();
00230
00231       return sendAndHandle("POST " + path, req, (status, responseBody) -> {
00232         if (status < 200 || status >= 300) {
00233           throw new ApiClientException("POST " + path + " failed with HTTP " + status + " body: " +
      responseBody);
00234         }
00235         return mapper.readValue(responseBody, responseType);
00236       });
00237     });
00238   }
00239
00240   // -------------------------------------------------------------------------
00241   // CRUD methods for Appetizers
00242   // -------------------------------------------------------------------------
00243
00251   public List<Appetizer> getAllAppetizers() {
00252     return getList("/api/appetizers", new TypeReference<List<Appetizer>>() {});
00253   }
00254
```

```
00263   public Appetizer getAppetizerById(Long id) {
00264     return get(API_APPETIZERS + id, Appetizer.class);
00265   }
00266
00274   public Appetizer createAppetizer(Appetizer appetizer) {
00275     return post("/api/appetizers", appetizer, Appetizer.class);
00276   }
00277
00286   public Appetizer updateAppetizer(Long id, Appetizer appetizer) {
00287     String path = API_APPETIZERS + id;
00288     return execute("updateAppetizer id=" + id, () -> {
00289       String json = mapper.writeValueAsString(appetizer);
00290       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00291           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
    APPLICATION_JSON).build();
00292
00293       return sendAndHandle("PUT " + path, req, (status, body) -> {
00294         if (status < 200 || status >= 300) {
00295           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
    body);
00296         }
00297         return mapper.readValue(body, Appetizer.class);
00298       });
00299     });
00300   }
00301
00308   public void deleteAppetizer(Long id) {
00309     String path = API_APPETIZERS + id;
00310     execute("deleteAppetizer id=" + id, () -> sendAndHandle("DELETE " + path,
00311         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).DELETE().build(), (status, body) -> {
00312           if (status < 200 || status >= 300) {
00313             throw new ApiClientException("DELETE " + path + " failed with HTTP " + status + " body: "
    + body);
00314           }
00315           return null;
00316         }));
00317   }
00318
00319   // ---------------------------------------------------------------------------
00320   // READ / UPDATE methods for Cashiers
00321   // ---------------------------------------------------------------------------
00322
00330   public List<Cashier> getAllCashiers() {
00331     return getList("/api/cashiers", new TypeReference<List<Cashier>>() {});
00332   }
00333
00341   public Cashier getCashierById(Long id) {
00342     return get("/api/cashiers/" + id, Cashier.class);
00343   }
00344
00352   public Cashier getCashierByName(String name) {
00353     return get("/api/cashiers/name/" + name, Cashier.class);
00354   }
00355
00365   public Cashier updateCashier(Long id, Cashier cashier) {
00366     String path = "/api/cashiers/" + id;
00367     return execute("updateCashier id=" + id, () -> {
00368       String json = mapper.writeValueAsString(cashier);
00369       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00370           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
    APPLICATION_JSON).build();
00371
00372       return sendAndHandle("PUT " + path, req, (status, body) -> {
00373         if (status < 200 || status >= 300) {
00374           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
    body);
00375         }
00376         return mapper.readValue(body, Cashier.class);
00377       });
00378     });
00379   }
00380
00381   // ---------------------------------------------------------------------------
00382   // CRUD methods for Drinks
00383   // ---------------------------------------------------------------------------
00384
00392   public List<Drink> getAllDrinks() {
00393     return getList("/api/drinks", new TypeReference<List<Drink>>() {});
00394   }
00395
00404   public Drink getDrinkById(Long id) {
00405     return get(API_DRINKS + id, Drink.class);
00406   }
00407
00416   public Drink createDrink(Drink drink) {
00417     return post("/api/drinks", drink, Drink.class);
00418   }
```

```
00419
00429   public Drink updateDrink(Long id, Drink drink) {
00430     String path = API_DRINKS + id;
00431     return execute("updateDrink id=" + id, () -> {
00432       String json = mapper.writeValueAsString(drink);
00433       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00434           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
      APPLICATION_JSON).build();
00435
00436       return sendAndHandle("PUT " + path, req, (status, body) -> {
00437         if (status < 200 || status >= 300) {
00438           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
      body);
00439         }
00440         return mapper.readValue(body, Drink.class);
00441       });
00442     });
00443   }
00444
00452   public void deleteDrink(Long id) {
00453     String path = API_DRINKS + id;
00454     execute("deleteDrink id=" + id, () -> sendAndHandle("DELETE " + path,
00455         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).DELETE().build(), (status, body) -> {
00456           if (status < 200 || status >= 300) {
00457             throw new ApiClientException("DELETE " + path + " failed with HTTP " + status + " body: "
      + body);
00458           }
00459           return null;
00460         }));
00461   }
00462
00463   // ------------------------------------------------------------------------
00464   // CRUD methods for MainCourses
00465   // ------------------------------------------------------------------------
00466
00474   public List<MainCourse> getAllMainCourses() {
00475     return getList("/api/maincourses", new TypeReference<List<MainCourse>>() {});
00476   }
00477
00486   public MainCourse getMainCourseById(Long id) {
00487     return get(API_MAINCOURSES + id, MainCourse.class);
00488   }
00489
00498   public MainCourse createMainCourse(MainCourse mainCourse) {
00499     return post("/api/maincourses", mainCourse, MainCourse.class);
00500   }
00501
00511   public MainCourse updateMainCourse(Long id, MainCourse mainCourse) {
00512     String path = API_MAINCOURSES + id;
00513     return execute("updateMainCourse id=" + id, () -> {
00514       String json = mapper.writeValueAsString(mainCourse);
00515       HttpRequest req = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00516           .PUT(HttpRequest.BodyPublishers.ofString(json)).header(CONTENT_TYPE,
      APPLICATION_JSON).build();
00517
00518       return sendAndHandle("PUT " + path, req, (status, body) -> {
00519         if (status < 200 || status >= 300) {
00520           throw new ApiClientException("PUT " + path + " failed with HTTP " + status + " body: " +
      body);
00521         }
00522         return mapper.readValue(body, MainCourse.class);
00523       });
00524     });
00525   }
00526
00534   public void deleteMainCourse(Long id) {
00535     String path = API_MAINCOURSES + id;
00536     execute("deleteMainCourse id=" + id, () -> sendAndHandle("DELETE " + path,
00537         HttpRequest.newBuilder().uri(URI.create(baseUrl + path)).DELETE().build(), (status, body) -> {
00538           if (status < 200 || status >= 300) {
00539             throw new ApiClientException("DELETE " + path + " failed with HTTP " + status + " body: "
      + body);
00540           }
00541           return null;
00542         }));
00543   }
00544
00545   // ------------------------------------------------------------------------
00546   // Authentication methods
00547   // ------------------------------------------------------------------------
00548
00556   public void login(String ignored) {
00557     // No-op: kept for compatibility.
00558   }
00559
00571   public User login(String username, String password) {
00572     String path = API_LOGIN;
```

```
00573     return execute("login for user=" + username, () -> {
00574       String jsonBody = mapper.writeValueAsString(Map.of(
00575           "username", username,
00576           "password", password));
00577
00578       HttpRequest request = HttpRequest.newBuilder().uri(URI.create(baseUrl + path))
00579           .header(CONTENT_TYPE,
      APPLICATION_JSON).POST(HttpRequest.BodyPublishers.ofString(jsonBody)).build();
00580
00581       return sendAndHandle("POST " + path, request, (status, body) -> {
00582         if (status == 200) {
00583           return mapper.readValue(body, User.class);
00584         } else {
00585           throw new ApiClientException("Login failed with status: " + status + " body: " + body);
00586         }
00587       });
00588     });
00589   }
00590 }
```

10.58 ApiClientException.java File Reference

Classes

- class es.ull.esit.app.middleware.ApiClientException

  Custom exception class for API client errors.

Packages

- package es.ull.esit.app.middleware

10.59 ApiClientException.java

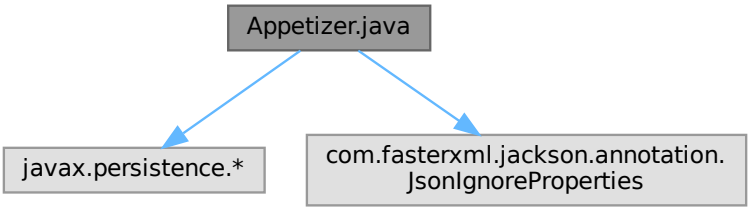
Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware;
00002
00009 public class ApiClientException extends RuntimeException {
00010
00016   public ApiClientException(String message) {
00017     super(message);
00018   }
00019
00026   public ApiClientException(String message, Throwable cause) {
00027     super(message, cause);
00028   }
00029 }
```

10.60 server/src/main/java/es/ull/esit/server/middleware/model/↩ Appetizer.java File Reference

```
import javax.persistence.*;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
```

Include dependency graph for server/src/main/java/es/ull/esit/server/middleware/model/Appetizer.java:



Classes

- class es.ull.esit.server.middleware.model.Appetizer

  JPA entity that represents an appetizer in the menu.

Packages

- package es.ull.esit.server.middleware.model

10.61 server/src/main/java/es/ull/esit/server/middleware/model/↩ Appetizer.java

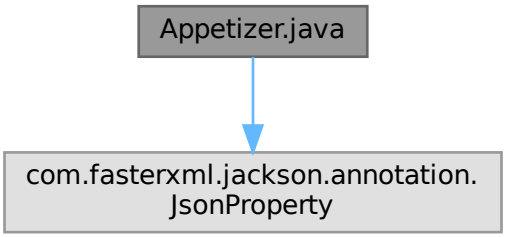
Go to the documentation of this file.

```
00001 package es.ull.esit.server.middleware.model;
00002
00003 import javax.persistence.*;
00004 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
00005
00012 @Entity
00013 @Table(name = "appetizers")
00014 @JsonIgnoreProperties({ "hibernateLazyInitializer", "handler" })
00015 public class Appetizer {
00016
00018   @Id
00019   @GeneratedValue(strategy = GenerationType.IDENTITY)
00020   @Column(name = "id")
00021   private Long appetizersId;
00022
00024   @Column(name = "name", nullable = false)
00025   private String itemAppetizers;
00026
00028   @Column(name = "price", nullable = false)
00029   private Integer appetizersPrice;
00030
00034   public Appetizer() {
00035   }
00036
00044   public Appetizer(Long appetizersId, String itemAppetizers, Integer appetizersPrice) {
00045     this.appetizersId = appetizersId;
00046     this.itemAppetizers = itemAppetizers;
00047     this.appetizersPrice = appetizersPrice;
00048   }
00049
00055   public Long getAppetizersId() {
00056     return appetizersId;
00057   }
00058
00065   public void setAppetizersId(Long appetizersId) {
```

```
00066     this.appetizersId = appetizersId;
00067   }
00068
00074   public String getItemAppetizers() {
00075     return itemAppetizers;
00076   }
00077
00083   public void setItemAppetizers(String itemAppetizers) {
00084     this.itemAppetizers = itemAppetizers;
00085   }
00086
00092   public Integer getAppetizersPrice() {
00093     return appetizersPrice;
00094   }
00095
00101   public void setAppetizersPrice(Integer appetizersPrice) {
00102     this.appetizersPrice = appetizersPrice;
00103   }
00104 }
```

10.62 src/main/java/es/ull/esit/app/middleware/model/Appetizer.java File Reference

`import com.fasterxml.jackson.annotation.JsonProperty;`
Include dependency graph for src/main/java/es/ull/esit/app/middleware/model/Appetizer.java:



Classes

- class es.ull.esit.app.middleware.model.Appetizer

  Client-side model representing an appetizer received from the backend API.

Packages

- package es.ull.esit.app.middleware.model

10.63 src/main/java/es/ull/esit/app/middleware/model/Appetizer.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.model;
00002
00003 import com.fasterxml.jackson.annotation.JsonProperty;
00004
00011 public class Appetizer {
00012
00014   @JsonProperty("appetizersId")
00015   private Long appetizersId;
00016
00018   @JsonProperty("itemAppetizers")
00019   private String itemAppetizers;
00020
00022   @JsonProperty("appetizersPrice")
00023   private Integer appetizersPrice;
00024
00029   @JsonProperty("receiptId")
00030   private Long receiptId;
00031
00035   public Appetizer() {
00036   }
00037
00046   public Appetizer(Long appetizersId, String itemAppetizers, Integer appetizersPrice, Long receiptId)
      {
00047     this.appetizersId = appetizersId;
00048     this.itemAppetizers = itemAppetizers;
00049     this.appetizersPrice = appetizersPrice;
00050     this.receiptId = receiptId;
00051   }
00052
00058   public Long getAppetizersId() {
00059     return appetizersId;
00060   }
00061
00067   public void setAppetizersId(Long appetizersId) {
00068     this.appetizersId = appetizersId;
00069   }
00070
00076   public String getItemAppetizers() {
00077     return itemAppetizers;
00078   }
00079
00085   public void setItemAppetizers(String itemAppetizers) {
00086     this.itemAppetizers = itemAppetizers;
00087   }
00088
00094   public Integer getAppetizersPrice() {
00095     return appetizersPrice;
00096   }
00097
00103   public void setAppetizersPrice(Integer appetizersPrice) {
00104     this.appetizersPrice = appetizersPrice;
00105   }
00106
00112   public Long getReceiptId() {
00113     return receiptId;
00114   }
00115
00121   public void setReceiptId(Long receiptId) {
00122     this.receiptId = receiptId;
00123   }
00124 }
```

10.64 BillResult.java File Reference

Classes

- class es.ull.esit.app.middleware.model.BillResult

  Data Transfer Object representing the result of a bill calculation.

Packages

- package es.ull.esit.app.middleware.model

10.65 BillResult.java

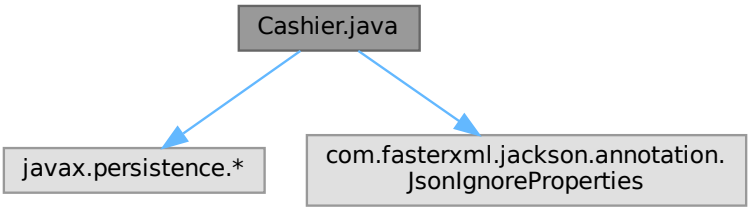
Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.model;
00002
00009 public class BillResult {
00010
00012   private final double subTotal;
00013
00015   private final double vat;
00016
00018   private final double total;
00019
00027   public BillResult(double subTotal, double vat, double total) {
00028     this.subTotal = subTotal;
00029     this.vat = vat;
00030     this.total = total;
00031   }
00032
00038   public double getSubTotal() {
00039     return subTotal;
00040   }
00041
00047   public double getVat() {
00048     return vat;
00049   }
00050
00056   public double getTotal() {
00057     return total;
00058   }
00059 }
```

10.66 server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java File Reference

```
import javax.persistence.*;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
```

Include dependency graph for server/src/main/java/es/ull/esit/server/middleware/model/Cashier.java:



Classes

- class es.ull.esit.server.middleware.model.Cashier

  JPA entity that represents a cashier in the system.

Packages

- package es.ull.esit.server.middleware.model

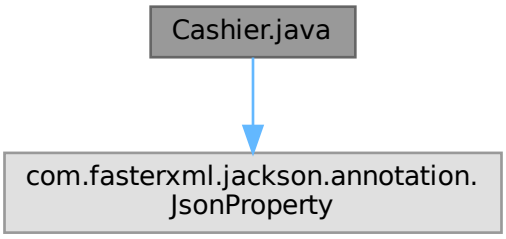
10.67 server/src/main/java/es/ull/esit/server/middleware/model/↩ Cashier.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.middleware.model;
00002
00003 import javax.persistence.*;
00004 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
00005
00012 @Entity
00013 @Table(name = "cashier")
00014 @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
00015 public class Cashier {
00016
00018   @Id
00019   @Column(name = "cashier_id")
00020   private Long id;
00021
00023   @Column(name = "cashier_name")
00024   private String name;
00025
00027   @Column(name = "cashier_salary")
00028   private Integer salary;
00029
00033   public Cashier() {
00034   }
00035
00043   public Cashier(Long id, String name, Integer salary) {
00044     this.id = id;
00045     this.name = name;
00046     this.salary = salary;
00047   }
00048
00054   public Long getId() {
00055     return id;
00056   }
00057
00064   public void setId(Long id) {
00065     this.id = id;
00066   }
00067
00073   public String getName() {
00074     return name;
00075   }
00076
00082   public void setName(String name) {
00083     this.name = name;
00084   }
00085
00091   public Integer getSalary() {
00092     return salary;
00093   }
00094
00100   public void setSalary(Integer salary) {
00101     this.salary = salary;
00102   }
00103 }
```

10.68 src/main/java/es/ull/esit/app/middleware/model/Cashier.java File Reference

`import com.fasterxml.jackson.annotation.JsonProperty;`
Include dependency graph for src/main/java/es/ull/esit/app/middleware/model/Cashier.java:



Classes

- class es.ull.esit.app.middleware.model.Cashier

  Client-side model representing a cashier returned by the backend.

Packages

- package es.ull.esit.app.middleware.model

10.69 src/main/java/es/ull/esit/app/middleware/model/Cashier.java

Go to the documentation of this file.

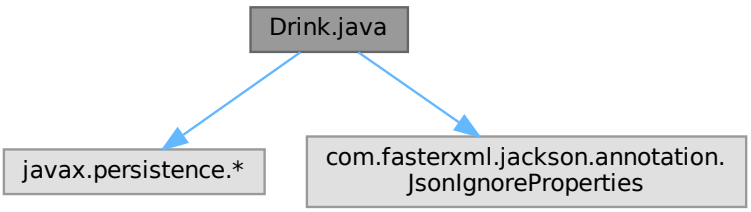
```
00001 package es.ull.esit.app.middleware.model;
00002
00003 import com.fasterxml.jackson.annotation.JsonProperty;
00004
00011 public class Cashier {
00012
00014   @JsonProperty("id")
00015   private Long id;
00016
00018   @JsonProperty("name")
00019   private String name;
00020
00022   @JsonProperty("salary")
00023   private Integer salary;
00024
00028   public Cashier() {
00029   }
00030
00038   public Cashier(Long id, String name, Integer salary) {
00039     this.id = id;
00040     this.name = name;
00041     this.salary = salary;
00042   }
00043
00049   public Long getId() {
00050     return id;
00051   }
```

```
00052
00058   public void setId(Long id) {
00059     this.id = id;
00060   }
00061
00067   public String getName() {
00068     return name;
00069   }
00070
00076   public void setName(String name) {
00077     this.name = name;
00078   }
00079
00085   public Integer getSalary() {
00086     return salary;
00087   }
00088
00094   public void setSalary(Integer salary) {
00095     this.salary = salary;
00096   }
00097 }
```

10.70 server/src/main/java/es/ull/esit/server/middleware/model/↩ Drink.java File Reference

```
import javax.persistence.*;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
```

Include dependency graph for server/src/main/java/es/ull/esit/server/middleware/model/Drink.java:



Classes

- class es.ull.esit.server.middleware.model.Drink

  JPA entity that represents a drink in the menu.

Packages

- package es.ull.esit.server.middleware.model

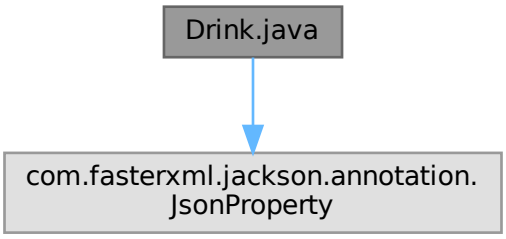
10.71 server/src/main/java/es/ull/esit/server/middleware/model/↩ Drink.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.middleware.model;
00002
00003 import javax.persistence.*;
00004 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
00005
00012 @Entity
00013 @Table(name = "drinks")
00014 @JsonIgnoreProperties({ "hibernateLazyInitializer", "handler" })
00015 public class Drink {
00016
00018   @Id
00019   @GeneratedValue(strategy = GenerationType.IDENTITY)
00020   @Column(name = "id")
00021   private Long drinksId;
00022
00024   @Column(name = "name", nullable = false)
00025   private String itemDrinks;
00026
00028   @Column(name = "price", nullable = false)
00029   private Integer drinksPrice;
00030
00034   public Drink() {
00035   }
00036
00044   public Drink(Long drinksId, String itemDrinks, Integer drinksPrice) {
00045     this.drinksId = drinksId;
00046     this.itemDrinks = itemDrinks;
00047     this.drinksPrice = drinksPrice;
00048   }
00049
00055   public Long getDrinksId() {
00056     return drinksId;
00057   }
00058
00066   public void setDrinksId(Long drinksId) {
00067     this.drinksId = drinksId;
00068   }
00069
00075   public String getItemDrinks() {
00076     return itemDrinks;
00077   }
00078
00084   public void setItemDrinks(String itemDrinks) {
00085     this.itemDrinks = itemDrinks;
00086   }
00087
00093   public Integer getDrinksPrice() {
00094     return drinksPrice;
00095   }
00096
00102   public void setDrinksPrice(Integer drinksPrice) {
00103     this.drinksPrice = drinksPrice;
00104   }
00105 }
```

10.72 src/main/java/es/ull/esit/app/middleware/model/Drink.java File Reference

`import com.fasterxml.jackson.annotation.JsonProperty;`
Include dependency graph for src/main/java/es/ull/esit/app/middleware/model/Drink.java:



Classes

- class es.ull.esit.app.middleware.model.Drink

  Client-side model representing a drink returned by the backend API.

Packages

- package es.ull.esit.app.middleware.model

10.73 src/main/java/es/ull/esit/app/middleware/model/Drink.java

Go to the documentation of this file.

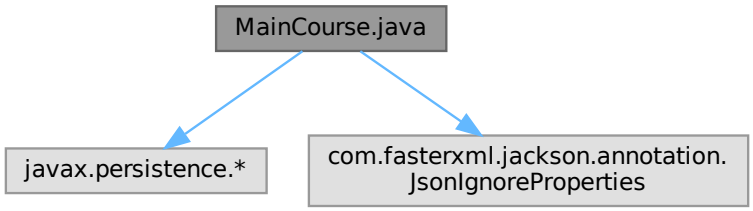
```
00001 package es.ull.esit.app.middleware.model;
00002
00003 import com.fasterxml.jackson.annotation.JsonProperty;
00004
00011 public class Drink {
00012
00014   @JsonProperty("drinksId")
00015   private Long drinksId;
00016
00018   @JsonProperty("itemDrinks")
00019   private String itemDrinks;
00020
00022   @JsonProperty("drinksPrice")
00023   private Integer drinksPrice;
00024
00029   @JsonProperty("receiptId")
00030   private Long receiptId;
00031
00035   public Drink() {
00036   }
00037
00046   public Drink(Long drinksId, String itemDrinks, Integer drinksPrice, Long receiptId) {
00047     this.drinksId = drinksId;
00048     this.itemDrinks = itemDrinks;
00049     this.drinksPrice = drinksPrice;
00050     this.receiptId = receiptId;
00051   }
```

```
00052
00058   public Long getDrinksId() {
00059     return drinksId;
00060   }
00061
00067   public void setDrinksId(Long drinksId) {
00068     this.drinksId = drinksId;
00069   }
00070
00076   public String getItemDrinks() {
00077     return itemDrinks;
00078   }
00079
00085   public void setItemDrinks(String itemDrinks) {
00086     this.itemDrinks = itemDrinks;
00087   }
00088
00094   public Integer getDrinksPrice() {
00095     return drinksPrice;
00096   }
00097
00103   public void setDrinksPrice(Integer drinksPrice) {
00104     this.drinksPrice = drinksPrice;
00105   }
00106
00112   public Long getReceiptId() {
00113     return receiptId;
00114   }
00115
00121   public void setReceiptId(Long receiptId) {
00122     this.receiptId = receiptId;
00123   }
00124 }
```

10.74 server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java File Reference

```
import javax.persistence.*;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
```

Include dependency graph for server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java:



Classes

- class es.ull.esit.server.middleware.model.MainCourse

  JPA entity that represents a main course in the menu.

Packages

- package es.ull.esit.server.middleware.model

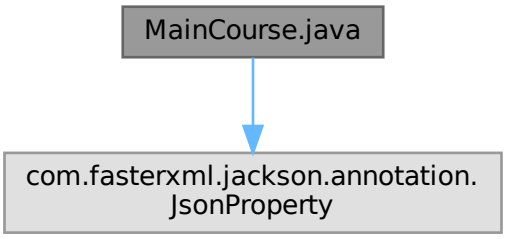
10.75 server/src/main/java/es/ull/esit/server/middleware/model/MainCourse.java

Go to the documentation of this file.

```
00001 package es.ull.esit.server.middleware.model;
00002
00003 import javax.persistence.*;
00004 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
00005
00012 @Entity
00013 @Table(name = "maincourse")
00014 @JsonIgnoreProperties({ "hibernateLazyInitializer", "handler" })
00015 public class MainCourse {
00016
00018   @Id
00019   @GeneratedValue(strategy = GenerationType.IDENTITY)
00020   @Column(name = "id")
00021   private Long foodId;
00022
00024   @Column(name = "name", nullable = false)
00025   private String itemFood;
00026
00028   @Column(name = "price", nullable = false)
00029   private Integer foodPrice;
00030
00034   public MainCourse() {
00035   }
00036
00044   public MainCourse(Long foodId, String itemFood, Integer foodPrice) {
00045     this.foodId = foodId;
00046     this.itemFood = itemFood;
00047     this.foodPrice = foodPrice;
00048   }
00049
00055   public Long getFoodId() {
00056     return foodId;
00057   }
00058
00066   public void setFoodId(Long foodId) {
00067     this.foodId = foodId;
00068   }
00069
00075   public String getItemFood() {
00076     return itemFood;
00077   }
00078
00084   public void setItemFood(String itemFood) {
00085     this.itemFood = itemFood;
00086   }
00087
00093   public Integer getFoodPrice() {
00094     return foodPrice;
00095   }
00096
00102   public void setFoodPrice(Integer foodPrice) {
00103     this.foodPrice = foodPrice;
00104   }
00105 }
```

10.76 src/main/java/es/ull/esit/app/middleware/model/MainCourse.java File Reference

`import com.fasterxml.jackson.annotation.JsonProperty;`
Include dependency graph for src/main/java/es/ull/esit/app/middleware/model/MainCourse.java:



Classes

- class es.ull.esit.app.middleware.model.MainCourse

  Client-side model representing a main course returned by the backend.

Packages

- package es.ull.esit.app.middleware.model

10.77 src/main/java/es/ull/esit/app/middleware/model/MainCourse.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.model;
00002
00003 import com.fasterxml.jackson.annotation.JsonProperty;
00004
00011 public class MainCourse {
00012
00014   @JsonProperty("foodId")
00015   private Long foodId;
00016
00018   @JsonProperty("itemFood")
00019   private String itemFood;
00020
00022   @JsonProperty("foodPrice")
00023   private Integer foodPrice;
00024
00029   @JsonProperty("receiptId")
00030   private Long receiptId;
00031
00035   public MainCourse() {
00036   }
00037
00046   public MainCourse(Long foodId, String itemFood, Integer foodPrice, Long receiptId) {
00047     this.foodId = foodId;
00048     this.itemFood = itemFood;
00049     this.foodPrice = foodPrice;
00050     this.receiptId = receiptId;
00051   }
```

```
00052
00058   public Long getFoodId() {
00059     return foodId;
00060   }
00061
00067   public void setFoodId(Long foodId) {
00068     this.foodId = foodId;
00069   }
00070
00076   public String getItemFood() {
00077     return itemFood;
00078   }
00079
00085   public void setItemFood(String itemFood) {
00086     this.itemFood = itemFood;
00087   }
00088
00094   public Integer getFoodPrice() {
00095     return foodPrice;
00096   }
00097
00103   public void setFoodPrice(Integer foodPrice) {
00104     this.foodPrice = foodPrice;
00105   }
00106
00112   public Long getReceiptId() {
00113     return receiptId;
00114   }
00115
00121   public void setReceiptId(Long receiptId) {
00122     this.receiptId = receiptId;
00123   }
00124 }
```

10.78 server/src/main/java/es/ull/esit/server/middleware/model/User.java File Reference

```
import javax.persistence.Entity;
import javax.persistence.Table;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Column;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonIgnore;
```

Include dependency graph for server/src/main/java/es/ull/esit/server/middleware/model/User.java:

Classes

- class es.ull.esit.server.middleware.model.User

  JPA entity that represents a user in the system.

Packages

- package es.ull.esit.server.middleware.model

10.79 server/src/main/java/es/ull/esit/server/middleware/model/User.java

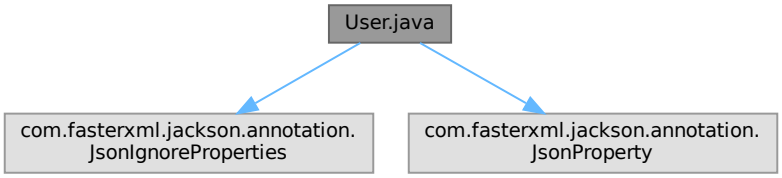
Go to the documentation of this file.

```
00001 package es.ull.esit.server.middleware.model;
00002
00003 import javax.persistence.Entity;
00004 import javax.persistence.Table;
00005 import javax.persistence.Id;
00006 import javax.persistence.GeneratedValue;
00007 import javax.persistence.GenerationType;
00008 import javax.persistence.Column;
00009
00010 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
00011 import com.fasterxml.jackson.annotation.JsonIgnore;
00012
00019 @Entity
00020 @Table(name = "users")
00021 @JsonIgnoreProperties({ "hibernateLazyInitializer", "handler" })
00022 public class User {
00023
00025   @Id
00026   @GeneratedValue(strategy = GenerationType.IDENTITY)
00027   private Long id;
00028
00030   @Column(unique = true, nullable = false)
00031   private String username;
00032
00034   @Column(name = "password_hash", nullable = false)
00035   @JsonIgnore
00036   private String passwordHash;
00037
00039   private String role;
00040
00046   public Long getId() {
00047     return id;
00048   }
00049
00055   public void setId(Long id) {
00056     this.id = id;
00057   }
00058
00064   public String getUsername() {
00065     return username;
00066   }
00067
00073   public void setUsername(String username) {
00074     this.username = username;
00075   }
00076
00082   public String getPasswordHash() {
00083     return passwordHash;
00084   }
00085
00091   public void setPasswordHash(String passwordHash) {
00092     this.passwordHash = passwordHash;
00093   }
00094
00100   public String getRole() {
00101     return role;
00102   }
00103
00109   public void setRole(String role) {
00110     this.role = role;
00111   }
00112 }
```

10.80 src/main/java/es/ull/esit/app/middleware/model/User.java File Reference

```
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
```

Include dependency graph for src/main/java/es/ull/esit/app/middleware/model/User.java:



Classes

- class es.ull.esit.app.middleware.model.User

  Client-side model representing an authenticated user.

Packages

- package es.ull.esit.app.middleware.model

10.81 src/main/java/es/ull/esit/app/middleware/model/User.java

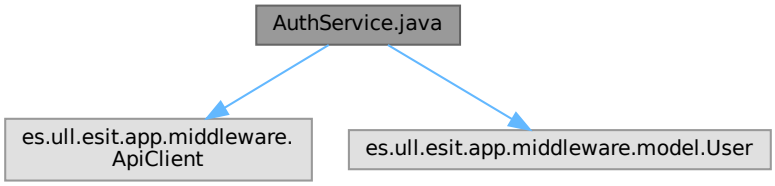
Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.model;
00002
00003 import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
00004 import com.fasterxml.jackson.annotation.JsonProperty;
00005
00012 @JsonIgnoreProperties(ignoreUnknown = true)
00013 public class User {
00014
00016   @JsonProperty("username")
00017   private String username;
00018
00020   @JsonProperty("role")
00021   private String role;
00022
00026   public User() {
00027   }
00028
00035   public User(String username, String role) {
00036     this.username = username;
00037     this.role = role;
00038   }
00039
00045   public String getUsername() {
00046     return username;
00047   }
00048
00054   public void setUsername(String username) {
00055     this.username = username;
00056   }
00057
00063   public String getRole() {
00064     return role;
00065   }
00066
00072   public void setRole(String role) {
00073     this.role = role;
00074   }
00075
00081   public boolean isAdmin() {
00082     return "ADMIN".equalsIgnoreCase(this.role);
00083   }
00084 }
```

10.82 AuthService.java File Reference

```
import es.ull.esit.app.middleware.ApiClient;
import es.ull.esit.app.middleware.model.User;
```

Include dependency graph for AuthService.java:



Classes

- class es.ull.esit.app.middleware.service.AuthService

  Service responsible for handling authentication logic on the client side.

Packages

- package es.ull.esit.app.middleware.service

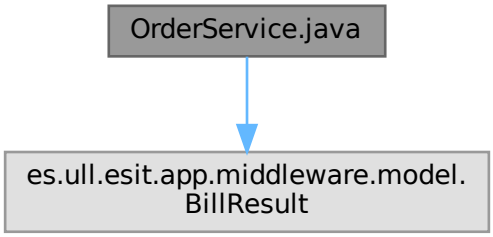
10.83 AuthService.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.service;
00002
00003 import es.ull.esit.app.middleware.ApiClient;
00004 import es.ull.esit.app.middleware.model.User;
00005
00012 public class AuthService {
00013
00015   private final ApiClient client;
00016
00022   public AuthService(ApiClient client) {
00023     this.client = client;
00024   }
00025
00037   public User authenticate(String username, String password) {
00038     // 1. Local Validation.
00039     if (username == null || username.trim().isEmpty()) {
00040       throw new IllegalArgumentException("Username cannot be empty.");
00041     }
00042     if (password == null || password.isEmpty()) {
00043       throw new IllegalArgumentException("Password cannot be empty.");
00044     }
00045
00046     // 2. Call API.
00047     return client.login(username, password);
00048   }
00049 }
```

10.84 OrderService.java File Reference

`import es.ull.esit.app.middleware.model.BillResult;`
Include dependency graph for OrderService.java:



Classes

- class es.ull.esit.app.middleware.service.OrderService

  Service that handles order-related calculations and receipt generation.

Packages

- package es.ull.esit.app.middleware.service

10.85 OrderService.java

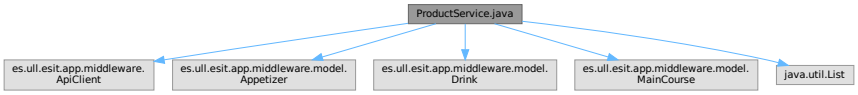
Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.service;
00002
00003 import es.ull.esit.app.middleware.model.BillResult;
00004
00011 public class OrderService {
00012
00014   private static final double VAT_RATE = 0.15;
00015
00024   public BillResult calculateBill(double itemsTotalSum) {
00025     double vat = itemsTotalSum * VAT_RATE;
00026     double total = itemsTotalSum + vat;
00027
00028     return new BillResult(
00029         Math.round(itemsTotalSum * 100.0) / 100.0,
00030         Math.round(vat * 100.0) / 100.0,
00031         Math.round(total * 100.0) / 100.0);
00032   }
00033
00044   public void generateReceiptFile(int receiptNo, BillResult bill) throws java.io.FileNotFoundException
      {
00045     // Ensure the directory exists.
00046     new java.io.File("receipts").mkdirs();
00047
00048     try (java.io.PrintWriter output = new java.io.PrintWriter("receipts/billNo." + receiptNo +
      ".txt")) {
00049       output.println(" Bill number is: " + receiptNo);
00050       output.println("==============");
00051       output.println("--------------");
00052       output.println("Subtotal is: " + bill.getSubTotal() + " SR");
00053       output.println("vat: " + bill.getVat() + " SR");
00054       output.println("Total is: " + bill.getTotal() + " SR");
00055       output.println();
00056       output.println("THANK YOU FOR ORDERING");
00057     }
00058   }
00059 }
```

10.86 ProductService.java File Reference

```
import es.ull.esit.app.middleware.ApiClient;
import es.ull.esit.app.middleware.model.Appetizer;
import es.ull.esit.app.middleware.model.Drink;
import es.ull.esit.app.middleware.model.MainCourse;
import java.util.List;
```

Include dependency graph for ProductService.java:



Classes

- class es.ull.esit.app.middleware.service.ProductService

  Service handling business logic for menu products.

Packages

- package es.ull.esit.app.middleware.service

10.87 ProductService.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.service;
00002
00003 import es.ull.esit.app.middleware.ApiClient;
00004 import es.ull.esit.app.middleware.model.Appetizer;
00005 import es.ull.esit.app.middleware.model.Drink;
00006 import es.ull.esit.app.middleware.model.MainCourse;
00007 import java.util.List;
00008
00016 public class ProductService {
00017
00019   private final ApiClient client;
00020
00026   public ProductService(ApiClient client) {
00027     this.client = client;
00028   }
00029
00030   // ==================== DRINKS LOGIC ====================
00031
00037   public List<Drink> getAllDrinks() {
00038     return client.getAllDrinks();
00039   }
00040
00047   public Drink getDrinkById(Long id) {
00048     return client.getDrinkById(id);
00049   }
00050
00059   public void addDrink(String name, String priceStr) {
00060     int price = validateAndParsePrice(priceStr);
00061     validateName(name);
00062
00063     Drink newDrink = new Drink(null, name, price, null);
00064     client.createDrink(newDrink);
00065   }
00066
00074   public void updateDrink(Long id, String name, String priceStr) {
00075     if (id == null) {
```

```
00076       throw new IllegalArgumentException("No drink selected!");
00077     }
00078
00079     int price = validateAndParsePrice(priceStr);
00080     validateName(name);
00081
00082     Drink updatedDrink = new Drink(id, name, price, null);
00083     client.updateDrink(id, updatedDrink);
00084   }
00085
00086   // ==================== APPETIZERS LOGIC ====================
00087
00093   public List<Appetizer> getAllAppetizers() {
00094     return client.getAllAppetizers();
00095   }
00096
00103   public Appetizer getAppetizerById(Long id) {
00104     return client.getAppetizerById(id);
00105   }
00106
00115   public void addAppetizer(String name, String priceStr) {
00116     int price = validateAndParsePrice(priceStr);
00117     validateName(name);
00118
00119     Appetizer newAppetizer = new Appetizer(null, name, price, null);
00120     client.createAppetizer(newAppetizer);
00121   }
00122
00130   public void updateAppetizer(Long id, String name, String priceStr) {
00131     if (id == null) {
00132       throw new IllegalArgumentException("No appetizer selected!");
00133     }
00134
00135     int price = validateAndParsePrice(priceStr);
00136     validateName(name);
00137
00138     Appetizer updatedAppetizer = new Appetizer(id, name, price, null);
00139     client.updateAppetizer(id, updatedAppetizer);
00140   }
00141
00142   // ==================== MAIN COURSE LOGIC ====================
00143
00149   public List<MainCourse> getAllMainCourses() {
00150     return client.getAllMainCourses();
00151   }
00152
00159   public MainCourse getMainCourseById(Long id) {
00160     return client.getMainCourseById(id);
00161   }
00162
00171   public void addMainCourse(String name, String priceStr) {
00172     int price = validateAndParsePrice(priceStr);
00173     validateName(name);
00174
00175     MainCourse newMainCourse = new MainCourse(null, name, price, null);
00176     client.createMainCourse(newMainCourse);
00177   }
00178
00186   public void updateMainCourse(Long id, String name, String priceStr) {
00187     if (id == null) {
00188       throw new IllegalArgumentException("No main course selected!");
00189     }
00190
00191     int price = validateAndParsePrice(priceStr);
00192     validateName(name);
00193
00194     MainCourse updatedMainCourse = new MainCourse(id, name, price, null);
00195     client.updateMainCourse(id, updatedMainCourse);
00196   }
00197
00198   // ==================== VALIDATION HELPERS ====================
00199
00206   private void validateName(String name) {
00207     if (name == null || name.trim().isEmpty()) {
00208       throw new IllegalArgumentException("Item name cannot be empty.");
00209     }
00210   }
00211
00221   private int validateAndParsePrice(String priceStr) {
00222     if (priceStr == null || priceStr.trim().isEmpty()) {
00223       throw new IllegalArgumentException("Price cannot be empty.");
00224     }
00225     try {
00226       int price = Integer.parseInt(priceStr.trim());
00227       if (price < 0) {
00228         throw new IllegalArgumentException("Price cannot be negative.");
00229       }
```

```
00230       return price;
00231     } catch (NumberFormatException e) {
00232       throw new IllegalArgumentException("Price must be a valid whole number.");
00233     }
00234   }
00235 }
```

10.88 ReportService.java File Reference

```
import es.ull.esit.app.middleware.ApiClient;
import es.ull.esit.app.middleware.model.Appetizer;
import es.ull.esit.app.middleware.model.Cashier;
import es.ull.esit.app.middleware.model.Drink;
import es.ull.esit.app.middleware.model.MainCourse;
import java.util.List;
```

Include dependency graph for ReportService.java:

Classes

- class es.ull.esit.app.middleware.service.ReportService

  Service providing reporting and system status operations.

Packages

- package es.ull.esit.app.middleware.service

10.89 ReportService.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app.middleware.service;
00002
00003 import es.ull.esit.app.middleware.ApiClient;
00004 import es.ull.esit.app.middleware.model.Appetizer;
00005 import es.ull.esit.app.middleware.model.Cashier;
00006 import es.ull.esit.app.middleware.model.Drink;
00007 import es.ull.esit.app.middleware.model.MainCourse;
00008 import java.util.List;
00009
00016 public class ReportService {
00017
00019   private final ApiClient client;
00020
00026   public ReportService(ApiClient client) {
00027     this.client = client;
00028   }
00029
00037   public List<Cashier> getCashierInfo() {
00038     return client.getAllCashiers();
00039   }
00040
00049   public String checkMenuStatus() {
00050     // Fetch lists to verify connectivity.
00051     List<Appetizer> appetizers = client.getAllAppetizers();
00052     List<Drink> drinks = client.getAllDrinks();
00053     List<MainCourse> mainCourses = client.getAllMainCourses();
00054
00055     return String.format(
00056         "Menu System Online: %d appetizers, %d drinks, %d main courses available.",
00057         appetizers.size(), drinks.size(), mainCourses.size());
00058   }
00059 }
```

10.90 Order.java File Reference

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import es.ull.esit.app.middleware.ApiClient;
import es.ull.esit.app.middleware.model.Appetizer;
import es.ull.esit.app.middleware.model.BillResult;
import es.ull.esit.app.middleware.model.Drink;
import es.ull.esit.app.middleware.model.MainCourse;
import es.ull.esit.app.middleware.service.OrderService;
import es.ull.esit.app.middleware.service.ProductService;
import java.io.PrintWriter;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableModel;
```

Include dependency graph for Order.java:

Classes

- class es.ull.esit.app.Order

  Main menu window for taking customer orders.

Packages

- package es.ull.esit.app

10.91 Order.java

Go to the documentation of this file.

```
00001 package es.ull.esit.app;
00002
00003 import org.slf4j.Logger;
00004 import org.slf4j.LoggerFactory;
00005
00006 import es.ull.esit.app.middleware.ApiClient;
00007 import es.ull.esit.app.middleware.model.Appetizer;
00008 import es.ull.esit.app.middleware.model.BillResult;
00009 import es.ull.esit.app.middleware.model.Drink;
00010 import es.ull.esit.app.middleware.model.MainCourse;
00011 import es.ull.esit.app.middleware.service.OrderService;
00012 import es.ull.esit.app.middleware.service.ProductService;
00013
00014 import java.io.PrintWriter;
00015 import javax.swing.JOptionPane;
00016 import javax.swing.SwingUtilities;
00017 import javax.swing.table.DefaultTableModel;
00018
00031 public class Order extends javax.swing.JFrame {
00032
00034   private static final String PRIMARY_FONT = "Yu Gothic UI";
00035
00036   private static final String PRIMARY_FONT_LIGHT = "Yu Gothic UI Light";
00037
00039   private final transient ProductService productService;
00040
00042   private final transient OrderService orderService = new OrderService();
00043
```

```
00045   private static final Logger LOGGER = LoggerFactory.getLogger(Order.class);
00046
00048   private DefaultTableModel drinksModel;
00049   private DefaultTableModel appetizersModel;
00050   private DefaultTableModel mainsModel;
00051
00053   private transient BillResult lastBill;
00054
00056   double subTotal;
00058   double vat;
00060   double total;
00061
00063   int receiptNo = 1;
00064
00069   transient PrintWriter output;
00070
00077   public Order() {
00078     initComponents();
00079
00080     // Initialize the Service Layer.
00081     ApiClient client = new ApiClient("http://localhost:8080");
00082     this.productService = new ProductService(client);
00083
00084     // Initialize receipt number label.
00085     receiptNoLbl.setText("Receipt No. : " + receiptNo);
00086
00087     // Configure tables and models.
00088     setupTables();
00089
00090     // Load menu items from database via REST API.
00091     loadMenuFromDatabase();
00092   }
00093
00101   Order(es.ull.esit.app.middleware.service.ProductService productService, boolean loadMenu) {
00102     initComponents();
00103     this.productService = productService;
00104
00105     // Initialize receipt number label.
00106     receiptNoLbl.setText("Receipt No. : " + receiptNo);
00107
00108     // Configure tables and models.
00109     setupTables();
00110
00111     if (loadMenu) {
00112       loadMenuFromDatabase();
00113     }
00114   }
00115
00120   Order(es.ull.esit.app.middleware.service.ProductService productService) {
00121     this(productService, true);
00122   }
00123
00133   private void setupTables() {
00134     // DRINKS
00135     drinksModel = new DefaultTableModel(
00136         new Object[] { "ID", "Item", "Price (SR)", "Qty" }, 0) {
00137       @Override
00138       public boolean isCellEditable(int row, int column) {
00139         // Only Qty column is editable
00140         return column == 3;
00141       }
00142
00143       @Override
00144       public Class<?> getColumnClass(int columnIndex) {
00145         return switch (columnIndex) {
00146           case 0 -> Long.class; // ID
00147           case 2, 3 -> Integer.class; // Price, Qty
00148           default -> String.class;
00149         };
00150       }
00151     };
00152     drinksTable.setModel(drinksModel);
00153
00154     // APPETIZERS
00155     appetizersModel = new DefaultTableModel(
00156         new Object[] { "ID", "Item", "Price (SR)", "Qty" }, 0) {
00157       @Override
00158       public boolean isCellEditable(int row, int column) {
00159         return column == 3;
00160       }
00161
00162       @Override
00163       public Class<?> getColumnClass(int columnIndex) {
00164         return switch (columnIndex) {
00165           case 0 -> Long.class;
00166           case 2, 3 -> Integer.class;
00167           default -> String.class;
```

```
00168         };
00169       }
00170     };
00171     appetizersTable.setModel(appetizersModel);
00172
00173     // MAIN COURSES
00174     mainsModel = new DefaultTableModel(
00175         new Object[] { "ID", "Item", "Price (SR)", "Qty" }, 0) {
00176       @Override
00177       public boolean isCellEditable(int row, int column) {
00178         return column == 3;
00179       }
00180
00181       @Override
00182       public Class<?> getColumnClass(int columnIndex) {
00183         return switch (columnIndex) {
00184           case 0 -> Long.class;
00185           case 2, 3 -> Integer.class;
00186           default -> String.class;
00187         };
00188       }
00189     };
00190     mainsTable.setModel(mainsModel);
00191   }
00192
00197   private void loadMenuFromDatabase() {
00198     new Thread(() -> {
00199       try {
00200         java.util.List<Drink> drinks = productService.getAllDrinks();
00201         java.util.List<Appetizer> appetizers = productService.getAllAppetizers();
00202         java.util.List<MainCourse> mains = productService.getAllMainCourses();
00203
00204         SwingUtilities.invokeLater(() -> {
00205           fillDrinks(drinks);
00206           fillAppetizers(appetizers);
00207           fillMains(mains);
00208         });
00209
00210     } catch (Exception ex) {
00211       LOGGER.error("Error loading menu from backend", ex);
00212       SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog(
00213       this,
00214       "Error loading menu from backend:\n" + ex.getMessage(),
00215       "Menu loading error",
00216       JOptionPane.ERROR_MESSAGE));
00217     }
00218       }).start();
00219   }
00220
00226   private void fillDrinks(java.util.List<Drink> drinks) {
00227     drinksModel.setRowCount(0);
00228     for (Drink d : drinks) {
00229       drinksModel.addRow(new Object[] {
00230           d.getDrinksId(),
00231           d.getItemDrinks(),
00232           d.getDrinksPrice(),
00233           0 // initial quantity
00234       });
00235     }
00236   }
00237
00243   private void fillAppetizers(java.util.List<Appetizer> appetizers) {
00244     appetizersModel.setRowCount(0);
00245     for (Appetizer a : appetizers) {
00246       appetizersModel.addRow(new Object[] {
00247           a.getAppetizersId(),
00248           a.getItemAppetizers(),
00249           a.getAppetizersPrice(),
00250           0
00251       });
00252     }
00253   }
00254
00260   private void fillMains(java.util.List<MainCourse> mains) {
00261     mainsModel.setRowCount(0);
00262     for (MainCourse m : mains) {
00263       mainsModel.addRow(new Object[] {
00264           m.getFoodId(),
00265           m.getItemFood(),
00266           m.getFoodPrice(),
00267           0
00268       });
00269     }
00270   }
00271
00282   private double sumFromModel(DefaultTableModel model) {
00283     double sum = 0.0;
```

```
00284     for (int row = 0; row < model.getRowCount(); row++) {
00285       Object qtyObj = model.getValueAt(row, 3); // Qty
00286       Object priceObj = model.getValueAt(row, 2); // Price
00287 
00288       if (qtyObj == null || priceObj == null)
00289         continue;
00290 
00291       int qty;
00292       int price;
00293       try {
00294         qty = ((Number) qtyObj).intValue();
00295         price = ((Number) priceObj).intValue();
00296       } catch (ClassCastException e) {
00297         // Fallback in case something comes as String
00298         qty = Integer.parseInt(qtyObj.toString());
00299         price = Integer.parseInt(priceObj.toString());
00300       }
00301 
00302       if (qty > 0) {
00303         sum += qty * price;
00304       }
00305     }
00306     return sum;
00307   }
00308 
00312   private void resetQtyColumn(DefaultTableModel model) {
00313     for (int row = 0; row < model.getRowCount(); row++) {
00314       model.setValueAt(0, row, 3); // column 3 is Qty
00315     }
00316   }
00317 
00324   void loadPrice() {
00325     // No-op
00326   }
00327 
00336   @SuppressWarnings("unchecked")
00337   // <editor-fold defaultstate="collapsed" desc="Generated
00338   // Code">//GEN-BEGIN:initComponents
00339   private void initComponents() {
00340 
00341     jPanel2 = new javax.swing.JPanel();
00342     drinksPnl = new javax.swing.JPanel();
00343     drinksScroll = new javax.swing.JScrollPane();
00344     drinksTable = new javax.swing.JTable();
00345     appetizerPnl = new javax.swing.JPanel();
00346     appetizersScroll = new javax.swing.JScrollPane();
00347     appetizersTable = new javax.swing.JTable();
00348     mainCoursePnl = new javax.swing.JPanel();
00349     mainsScroll = new javax.swing.JScrollPane();
00350     mainsTable = new javax.swing.JTable();
00351     jPanel1 = new javax.swing.JPanel();
00352     subTotalLbl = new javax.swing.JLabel();
00353     vatLbl = new javax.swing.JLabel();
00354     totalLbl = new javax.swing.JLabel();
00355     receiptNoLbl = new javax.swing.JLabel();
00356     payBtn = new javax.swing.JButton();
00357     newReceiptBtn = new javax.swing.JButton();
00358     saveReceiptBtn = new javax.swing.JButton();
00359     jLabel1 = new javax.swing.JLabel();
00360     goBackMenueBtn = new javax.swing.JButton();
00361     jLabel2 = new javax.swing.JLabel();
00362 
00363     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
00364     setTitle("Menu");
00365     setBackground(new java.awt.Color(204, 204, 204));
00366     setResizable(false);
00367 
00368     jPanel2.setBackground(new java.awt.Color(248, 244, 230));
00369 
00370     drinksPnl.setBackground(new java.awt.Color(248, 244, 230));
00371     drinksPnl.setBorder(javax.swing.BorderFactory.createTitledBorder(
00372         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Drinks",
00373         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00374         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00375     drinksPnl.setPreferredSize(new java.awt.Dimension(260, 278));
00376 
00377     drinksTable.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00378     drinksTable.setModel(new javax.swing.table.DefaultTableModel(
00379         new Object[][] {
00380 
00381         },
00382         new String[] {
00383 
00384         }));
00385     drinksTable.getTableHeader().setReorderingAllowed(false);
00386     drinksScroll.setViewportView(drinksTable);
00387 
```

```
00388     javax.swing.GroupLayout drinksPnlLayout = new javax.swing.GroupLayout(drinksPnl);
00389     drinksPnl.setLayout(drinksPnlLayout);
00390     drinksPnlLayout.setHorizontalGroup(
00391         drinksPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00392             .addGroup(drinksPnlLayout.createSequentialGroup()
00393                 .addContainerGap()
00394                 .addComponent(drinksScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 281,
      Short.MAX_VALUE)
00395                 .addContainerGap()));
00396     drinksPnlLayout.setVerticalGroup(
00397         drinksPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00398             .addGroup(drinksPnlLayout.createSequentialGroup()
00399                 .addContainerGap()
00400                 .addComponent(drinksScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 186,
      Short.MAX_VALUE)
00401                 .addContainerGap()));
00402
00403     appetizerPnl.setBackground(new java.awt.Color(248, 244, 230));
00404     appetizerPnl.setBorder(javax.swing.BorderFactory.createTitledBorder(
00405         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Appetizers",
00406         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00407         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00408     appetizerPnl.setPreferredSize(new java.awt.Dimension(260, 278));
00409
00410     appetizersTable.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00411     appetizersTable.setModel(new javax.swing.table.DefaultTableModel(
00412         new Object[][] {
00413
00414         },
00415         new String[] {
00416
00417         }));
00418     appetizersTable.getTableHeader().setReorderingAllowed(false);
00419     appetizersScroll.setViewportView(appetizersTable);
00420
00421     javax.swing.GroupLayout appetizerPnlLayout = new javax.swing.GroupLayout(appetizerPnl);
00422     appetizerPnl.setLayout(appetizerPnlLayout);
00423     appetizerPnlLayout.setHorizontalGroup(
00424         appetizerPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00425             .addGroup(appetizerPnlLayout.createSequentialGroup()
00426                 .addContainerGap()
00427                 .addComponent(appetizersScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 368,
      Short.MAX_VALUE)
00428                 .addContainerGap()));
00429     appetizerPnlLayout.setVerticalGroup(
00430         appetizerPnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00431             .addGroup(appetizerPnlLayout.createSequentialGroup()
00432                 .addContainerGap()
00433                 .addComponent(appetizersScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 186,
      Short.MAX_VALUE)
00434                 .addContainerGap()));
00435
00436     mainCoursePnl.setBackground(new java.awt.Color(248, 244, 230));
00437     mainCoursePnl.setBorder(javax.swing.BorderFactory.createTitledBorder(
00438         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Main Course",
00439         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00440         new java.awt.Font(PRIMARY_FONT, 0, 18))); // NOI18N
00441     mainCoursePnl.setPreferredSize(new java.awt.Dimension(260, 278));
00442
00443     mainsTable.setFont(new java.awt.Font(PRIMARY_FONT, 0, 14)); // NOI18N
00444     mainsTable.setModel(new javax.swing.table.DefaultTableModel(
00445         new Object[][] {
00446
00447         },
00448         new String[] {
00449
00450         }));
00451     mainsTable.getTableHeader().setReorderingAllowed(false);
00452     mainsScroll.setViewportView(mainsTable);
00453
00454     javax.swing.GroupLayout mainCoursePnlLayout = new javax.swing.GroupLayout(mainCoursePnl);
00455     mainCoursePnl.setLayout(mainCoursePnlLayout);
00456     mainCoursePnlLayout.setHorizontalGroup(
00457         mainCoursePnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00458             .addGroup(mainCoursePnlLayout.createSequentialGroup()
00459                 .addContainerGap()
00460                 .addComponent(mainsScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 427, Short.MAX_VALUE)
00461                 .addContainerGap()));
00462     mainCoursePnlLayout.setVerticalGroup(
00463         mainCoursePnlLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00464             .addGroup(mainCoursePnlLayout.createSequentialGroup()
00465                 .addContainerGap()
00466                 .addComponent(mainsScroll, javax.swing.GroupLayout.DEFAULT_SIZE, 186, Short.MAX_VALUE)
00467                 .addContainerGap()));
00468
00469     jPanel1.setBackground(new java.awt.Color(248, 244, 230));
00470     jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(
```

```
00471         new javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0), 1, true), "Receipt",
00472         javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP,
00473         new java.awt.Font(PRIMARY_FONT, 0, 14))); // NOI18N
00474 
00475     subTotalLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 18)); // NOI18N
00476     subTotalLbl.setText("SubTotal: 0.0 SR");
00477 
00478     vatLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 18)); // NOI18N
00479     vatLbl.setText("VAT included: 0.0 SR");
00480 
00481     totalLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 18)); // NOI18N
00482     totalLbl.setText("Total: 0.0 SR");
00483 
00484     receiptNoLbl.setFont(new java.awt.Font(PRIMARY_FONT_LIGHT, 0, 14)); // NOI18N
00485     receiptNoLbl.setText("Receipt No. : 0");
00486 
00487     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
00488     jPanel1.setLayout(jPanel1Layout);
00489     jPanel1Layout.setHorizontalGroup(
00490         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00491             .addGroup(jPanel1Layout.createSequentialGroup()
00492                 .addContainerGap()
00493                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00494                     .addComponent(subTotalLbl)
00495                     .addComponent(vatLbl)
00496                     .addComponent(totalLbl)
00497                     .addComponent(receiptNoLbl))
00498                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)));
00499     jPanel1Layout.setVerticalGroup(
00500         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00501             .addGroup(jPanel1Layout.createSequentialGroup()
00502                 .addContainerGap()
00503                 .addComponent(subTotalLbl)
00504                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00505                 .addComponent(vatLbl)
00506                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
00507                 .addComponent(totalLbl)
00508                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 10,
    Short.MAX_VALUE)
00509                 .addComponent(receiptNoLbl)));
00510 
00511     payBtn.setFont(new java.awt.Font("Lucida Grande", 1, 18)); // NOI18N
00512     payBtn.setText("Pay");
00513     payBtn.addActionListener(this::payBtnActionPerformed);
00514 
00515     newReceiptBtn.setFont(new java.awt.Font(PRIMARY_FONT, 1, 14)); // NOI18N
00516     newReceiptBtn.setText("New Receipt");
00517     newReceiptBtn.addActionListener(this::newReceiptBtnActionPerformed);
00518 
00519     saveReceiptBtn.setFont(new java.awt.Font(PRIMARY_FONT, 1, 14)); // NOI18N
00520     saveReceiptBtn.setText("Save Receipt");
00521     saveReceiptBtn.addActionListener(this::saveReceiptBtnActionPerformed);
00522 
00523     jLabel1.setBackground(new java.awt.Color(255, 153, 0));
00524     jLabel1.setFont(new java.awt.Font(PRIMARY_FONT, 1, 36)); // NOI18N
00525     jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
00526     jLabel1.setText("BLACK PLATE MENU");
00527 
00528     goBackMenueBtn.setFont(new java.awt.Font(PRIMARY_FONT, 1, 14)); // NOI18N
00529     goBackMenueBtn.setText("Go Back");
00530     goBackMenueBtn.addActionListener(this::goBackMenueBtnActionPerformed);
00531 
00532     jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ui/screenshot.png"))); //
    NOI18N
00533 
00534     javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
00535     jPanel2.setLayout(jPanel2Layout);
00536     jPanel2Layout.setHorizontalGroup(
00537         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00538             .addGroup(jPanel2Layout.createSequentialGroup()
00539                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00540                     .addGroup(jPanel2Layout.createSequentialGroup()
00541                         .addGap(40, 40, 40)
00542                         .addComponent(drinksPnl, javax.swing.GroupLayout.PREFERRED_SIZE, 305,
00543                             javax.swing.GroupLayout.PREFERRED_SIZE)
00544                         .addGap(18, 18, 18)
00545                         .addComponent(appetizerPnl, javax.swing.GroupLayout.PREFERRED_SIZE, 392,
00546                             javax.swing.GroupLayout.PREFERRED_SIZE)
00547                         .addGap(18, 18, 18)
00548                         .addComponent(mainCoursePnl, javax.swing.GroupLayout.PREFERRED_SIZE, 451,
00549                             javax.swing.GroupLayout.PREFERRED_SIZE))
00550                     .addGroup(jPanel2Layout.createSequentialGroup()
00551                         .addGap(251, 251, 251)
00552                         .addComponent(jLabel2)
00553                         .addGap(18, 18, 18)
00554                         .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 433,
00555                             javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
00556                     .addGroup(jPanel2Layout.createSequentialGroup()
00557                         .addGap(16, 16, 16)
00558                         .addComponent(goBackMenueBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 130,
00559                             javax.swing.GroupLayout.PREFERRED_SIZE)
00560                         .addGap(147, 147, 147)
00561                         .addComponent(payBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 180,
00562                             javax.swing.GroupLayout.PREFERRED_SIZE)
00563                         .addGap(18, 18, 18)
00564
      .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00565                         .addComponent(newReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 220,
00566                             javax.swing.GroupLayout.PREFERRED_SIZE)
00567                         .addComponent(saveReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 220,
00568                             javax.swing.GroupLayout.PREFERRED_SIZE))
00569                         .addGap(28, 28, 28)
00570                         .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
00571                             javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)))
00572                 .addContainerGap(30, Short.MAX_VALUE)));
00573     jPanel2Layout.setVerticalGroup(
00574         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00575             .addGroup(jPanel2Layout.createSequentialGroup()
00576                 .addContainerGap(16, Short.MAX_VALUE)
00577                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00578                     .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.TRAILING)
00579                     .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING,
00580                         javax.swing.GroupLayout.PREFERRED_SIZE, 60,
      javax.swing.GroupLayout.PREFERRED_SIZE))
00581                 .addGap(40, 40, 40)
00582                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
      false)
00583                     .addComponent(appetizerPnl, javax.swing.GroupLayout.DEFAULT_SIZE, 230,
      Short.MAX_VALUE)
00584                     .addComponent(mainCoursePnl, javax.swing.GroupLayout.DEFAULT_SIZE, 230,
      Short.MAX_VALUE)
00585                     .addComponent(drinksPnl, javax.swing.GroupLayout.DEFAULT_SIZE, 230,
      Short.MAX_VALUE))
00586                 .addGap(18, 18, 18)
00587                 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00588                     .addComponent(goBackMenueBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
00589                         javax.swing.GroupLayout.PREFERRED_SIZE)
00590                     .addGroup(jPanel2Layout.createSequentialGroup()
00591                         .addComponent(newReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
00592                             javax.swing.GroupLayout.PREFERRED_SIZE)
00593                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
00594                         .addComponent(saveReceiptBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
00595                             javax.swing.GroupLayout.PREFERRED_SIZE))
00596                     .addComponent(payBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
00597                         javax.swing.GroupLayout.PREFERRED_SIZE)
00598                     .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00599                         javax.swing.GroupLayout.PREFERRED_SIZE))
00600                 .addGap(30, 30, 30)));
00601
00602     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
00603     getContentPane().setLayout(layout);
00604     layout.setHorizontalGroup(
00605         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00606             .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00607                 Short.MAX_VALUE));
00608     layout.setVerticalGroup(
00609         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
00610             .addGroup(layout.createSequentialGroup()
00611                 .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
00612                     Short.MAX_VALUE)
00613                 .addContainerGap()));
00614
00615     pack();
00616     setLocationRelativeTo(null);
00617   }// </editor-fold>//GEN-END:initComponents
00618
00632   private void payBtnActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_payBtnActionPerformed
00633     double itemsTotal = 0.0;
00634
00635     itemsTotal += sumFromModel(drinksModel);
00636     itemsTotal += sumFromModel(appetizersModel);
00637     itemsTotal += sumFromModel(mainsModel);
00638
00639     if (itemsTotal == 0.0) {
00640       JOptionPane.showMessageDialog(
00641           this,
00642           "Please select at least one item (Qty > 0).",
00643           "No items selected",
00644           JOptionPane.INFORMATION_MESSAGE);
```

```
00645       return;
00646     }
00647
00648     // Calculate bill using the dedicated service
00649     lastBill = orderService.calculateBill(itemsTotal);
00650
00651     subTotal = lastBill.getSubTotal();
00652     vat = lastBill.getVat();
00653     total = lastBill.getTotal();
00654
00655     subTotalLbl.setText("SubTotal: " + subTotal + " SR");
00656     vatLbl.setText("VAT included: " + vat + " SR");
00657     totalLbl.setText("Total: " + total + " SR");
00658
00659     JOptionPane.showMessageDialog(this, "Paid successfully");
00660   }// GEN-LAST:event_payBtnActionPerformed
00661
00675   private void saveReceiptBtnActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_saveReceiptBtnActionPerformed
00676     try {
00677       if (lastBill == null || lastBill.getTotal() == 0.0) {
00678         JOptionPane.showMessageDialog(
00679             this,
00680             "There is no paid bill to save.\nPlease press Pay first.",
00681             "No bill",
00682             JOptionPane.INFORMATION_MESSAGE);
00683         return;
00684       }
00685
00686       orderService.generateReceiptFile(receiptNo, lastBill);
00687
00688       JOptionPane.showMessageDialog(
00689           this,
00690           "Receipt number: " + receiptNo + " has been saved successfully.",
00691           "Receipt saved",
00692           JOptionPane.INFORMATION_MESSAGE);
00693
00694     } catch (Exception ex) {
00695       LOGGER.error("Error saving receipt", ex);
00696       JOptionPane.showMessageDialog(
00697           this,
00698           "Error saving receipt:\n" + ex.getMessage(),
00699           "Error",
00700           JOptionPane.ERROR_MESSAGE);
00701     }
00702   }// GEN-LAST:event_saveReceiptBtnActionPerformed
00703
00717   private void newReceiptBtnActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_newReceiptBtnActionPerformed
00718     if (total == 0.0) {
00719       // Nothing to reset
00720       return;
00721     }
00722
00723     resetQtyColumn(drinksModel);
00724     resetQtyColumn(appetizersModel);
00725     resetQtyColumn(mainsModel);
00726
00727     subTotal = 0.0;
00728     vat = 0.0;
00729     total = 0.0;
00730     lastBill = null;
00731
00732     subTotalLbl.setText("SubTotal: 0.0 SR");
00733     vatLbl.setText("VAT included: 0.0 SR");
00734     totalLbl.setText("Total: 0.0 SR");
00735
00736     receiptNo++;
00737     receiptNoLbl.setText("Receipt No. : " + receiptNo);
00738   }// GEN-LAST:event_newReceiptBtnActionPerformed
00739
00749   private void goBackMenueBtnActionPerformed(java.awt.event.ActionEvent evt) {//
      GEN-FIRST:event_goBackMenueBtnActionPerformed
00750     this.dispose();
00751     new Login().setVisible(true);
00752   }// GEN-LAST:event_goBackMenueBtnActionPerformed
00753
00763   public static void main(String[] args) {
00764     /* Set the Nimbus look and feel */
00765     // <editor-fold defaultstate="collapsed" desc=" Look and feel setting code
00766     // (optional) ">
00767     /*
00768      * If Nimbus (introduced in Java SE 6) is not available, stay with the default
00769      * look and feel.
00770      * For details see
00771      * http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
00772      */
```

```
00773     try {
00774       for (javax.swing.UIManager.LookAndFeelInfo info :
      javax.swing.UIManager.getInstalledLookAndFeels()) {
00775         if ("Nimbus".equals(info.getName())) {
00776           javax.swing.UIManager.setLookAndFeel(info.getClassName());
00777           break;
00778         }
00779       }
00780     } catch (ClassNotFoundException | javax.swing.UnsupportedLookAndFeelException |
00781         InstantiationException | IllegalAccessException ex) {
00782       java.util.logging.Logger.getLogger(Order.class.getName()).log(java.util.logging.Level.SEVERE,
      null, ex);
00783     }
00784     // </editor-fold>
00785 
00786     /* Create and display the form */
00787     java.awt.EventQueue.invokeLater(() -> new Order().setVisible(true));
00788   }
00789 
00790   // Variables declaration - do not modify//GEN-BEGIN:variables
00791   // Sonarqube rule java:S1450 must be ignored here as these variables are
00792   // auto-generated by the Form Editor and need to remain as instance variables.
00794   private javax.swing.JPanel appetizerPnl;
00796   private javax.swing.JPanel drinksPnl;
00798   private javax.swing.JPanel mainCoursePnl;
00800   private javax.swing.JTable appetizersTable;
00802   private javax.swing.JScrollPane appetizersScroll;
00804   private javax.swing.JTable drinksTable;
00806   private javax.swing.JScrollPane drinksScroll;
00808   private javax.swing.JButton goBackMenueBtn;
00810   private javax.swing.JLabel jLabel1;
00812   private javax.swing.JLabel jLabel2;
00814   private javax.swing.JPanel jPanel1;
00816   private javax.swing.JPanel jPanel2;
00818   private javax.swing.JTable mainsTable;
00820   private javax.swing.JScrollPane mainsScroll;
00822   private javax.swing.JButton newReceiptBtn;
00824   private javax.swing.JButton payBtn;
00826   private javax.swing.JLabel receiptNoLbl;
00828   private javax.swing.JButton saveReceiptBtn;
00830   private javax.swing.JLabel subTotalLbl;
00832   private javax.swing.JLabel totalLbl;
00834   private javax.swing.JLabel vatLbl;
00835   // End of variables declaration//GEN-END:variables
00836 }
```

Index