



**ESCUELA POLITÉCNICA
NACIONAL**
ESCUELA DE FORMACIÓN DE TECNÓLOGOS



Programación Orientada a Objetos

ASIGNATURA:	Diseño de Interfaces
PROFESOR:	Ing. Lorena Chulde
PERÍODO ACADÉMICO:	2024-A

Proyecto Final Minimarket

2024-A

Luis Adrian Ramos Guzman



PLANIFICACIÓN

OBJETIVO

Desarrollar un sistema de gestión para un Minimarket que permita a los usuarios, en roles de administrador y cajero, gestionar transacciones de venta, administrar el inventario y realizar un seguimiento de las ventas.

ALCANCE

- **Login para acceso de administradores y cajeros.**
- **Gestión de transacciones de compra y generación de notas de venta en PDF.**
- **Reducción automática del stock de productos comprados.**
- **Visualización de imágenes de productos.**
- **Funcionalidades exclusivas para el administrador, como agregar productos a stock, revisar ventas y gestionar usuarios cajeros.**

ANÁLISIS DE REQUISITOS

Requisitos Funcionales

1. Login y Autenticación:

- El sistema debe permitir el login para administradores y cajeros.
- Debe verificar las credenciales y autenticar a los usuarios según su rol.

2. Gestión de Transacciones:

- El cajero debe poder realizar transacciones de compra.
- Cada transacción debe ser registrada junto con el cajero que la realizó.
- Al finalizar una transacción, se debe generar una nota de venta en PDF.

3. Manejo de Stock:

- Cuando se compre un producto, el sistema debe reducir automáticamente el stock disponible.
- El administrador debe poder agregar nuevos productos al stock.

4. Visualización de Productos:

- El sistema debe mostrar una imagen del producto al cajero durante la compra.

5. Gestión de Ventas:

- El administrador debe poder revisar todas las ventas realizadas.
- El administrador debe poder revisar las ventas realizadas por cada cajero individualmente.

6. Gestión de Usuarios:

- El administrador debe poder agregar nuevos usuarios cajeros.

Requisitos No Funcionales**1. Usabilidad:**

- El sistema debe tener una interfaz de usuario intuitiva y fácil de usar para administradores y cajeros.

1. Rendimiento:

- El sistema debe ser capaz de procesar transacciones rápidamente, con tiempos de respuesta inferiores a 2 segundos.

2. Seguridad:

- El sistema debe asegurar que solo usuarios autenticados puedan acceder a las funcionalidades.
- Los datos sensibles, como las contraseñas, deben ser almacenados de forma segura (ej. cifrado).

3. 4. Disponibilidad:

- El sistema debe estar disponible el 99.9% del tiempo, exceptuando el mantenimiento programado.

4. 5. Escalabilidad:

- El sistema debe ser capaz de manejar un aumento en el número de usuarios y transacciones sin degradar el rendimiento.

5. 6. Mantenibilidad:

- El código del sistema debe estar bien documentado y estructurado para facilitar su mantenimiento y futuras actualizaciones.

Requisitos de Usuario**1. Cajero:**

- Debe poder iniciar sesión en el sistema.
- Debe poder realizar transacciones de compra.
- Debe poder ver imágenes de los productos.
- Debe recibir una nota de venta en PDF al finalizar la transacción.

2. Administrador:

- Debe poder iniciar sesión en el sistema.
- Debe poder agregar productos al stock.
- Debe poder revisar las ventas realizadas por todos los cajeros.
- Debe poder revisar las ventas individualmente por cada cajero.
- Debe poder agregar nuevos usuarios cajeros.

Requisitos de Sistema

1. Base de Datos:

- El sistema debe utilizar una base de datos relacional para almacenar información de usuarios, productos, transacciones y stock.
- La base de datos debe ser capaz de manejar consultas concurrentes sin problemas de rendimiento.

2. Interfaz de Usuario:

- El sistema debe tener una interfaz web accesible desde navegadores modernos.

DISEÑO

ARQUITECTURA DEL SISTEMA

La arquitectura del sistema se basa en una estructura de capas que incluye:

- **Capa de Presentación:** Interfaces de usuario (UI) para el login, gestión de transacciones, visualización de productos y administración.
- **Capa de Lógica de Negocio:** Clases y métodos que implementan la lógica de la aplicación, como la gestión de transacciones y la administración del stock.
- **Capa de Persistencia:** Gestión de la base de datos para almacenar información de usuarios, productos, transacciones y stock.

DISEÑO DE LA BASE DE DATOS

DISEÑO DE DIAGRAMAS DE FLUJO Y CLASES

DISEÑOS DE INTERFACES

LOGIN.form x

Component Tree

Minimarket La T U C A

Inicio de Sesión

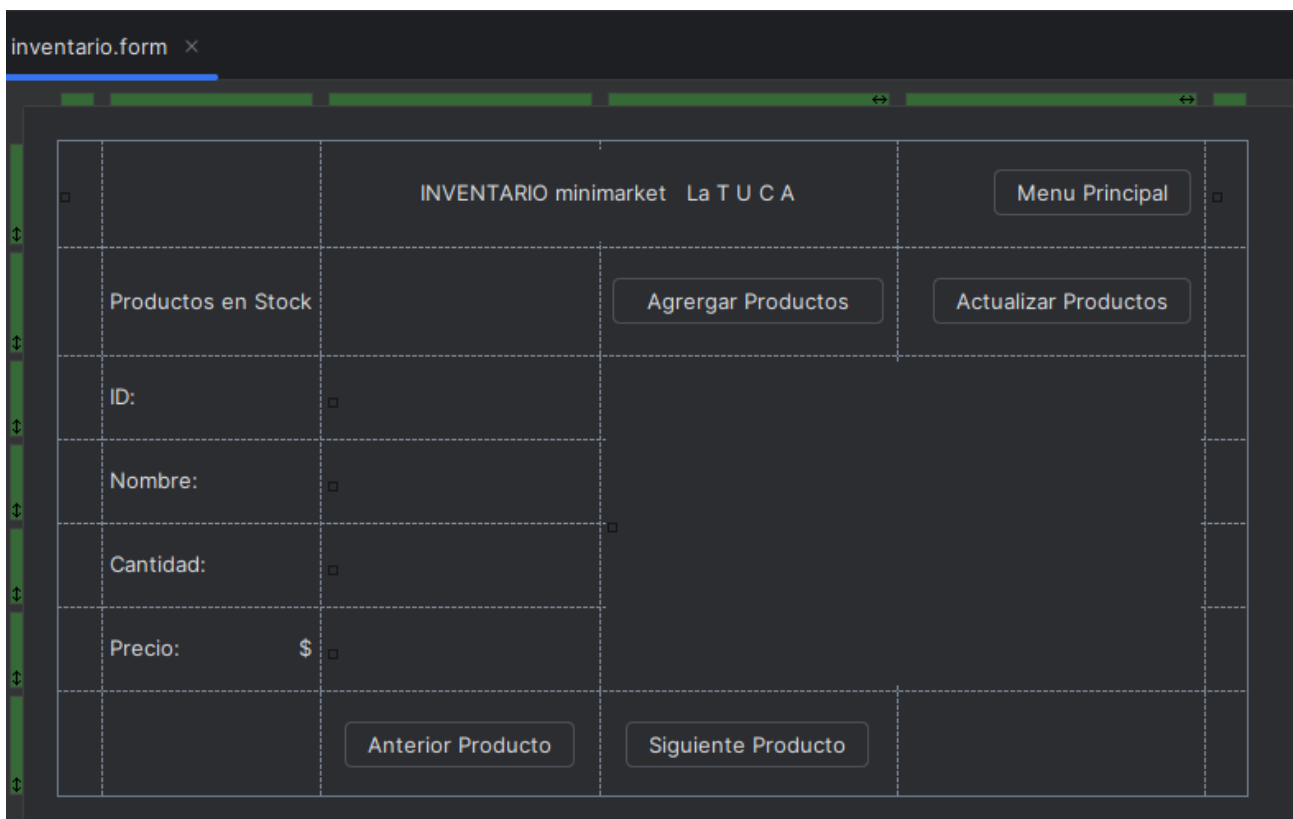
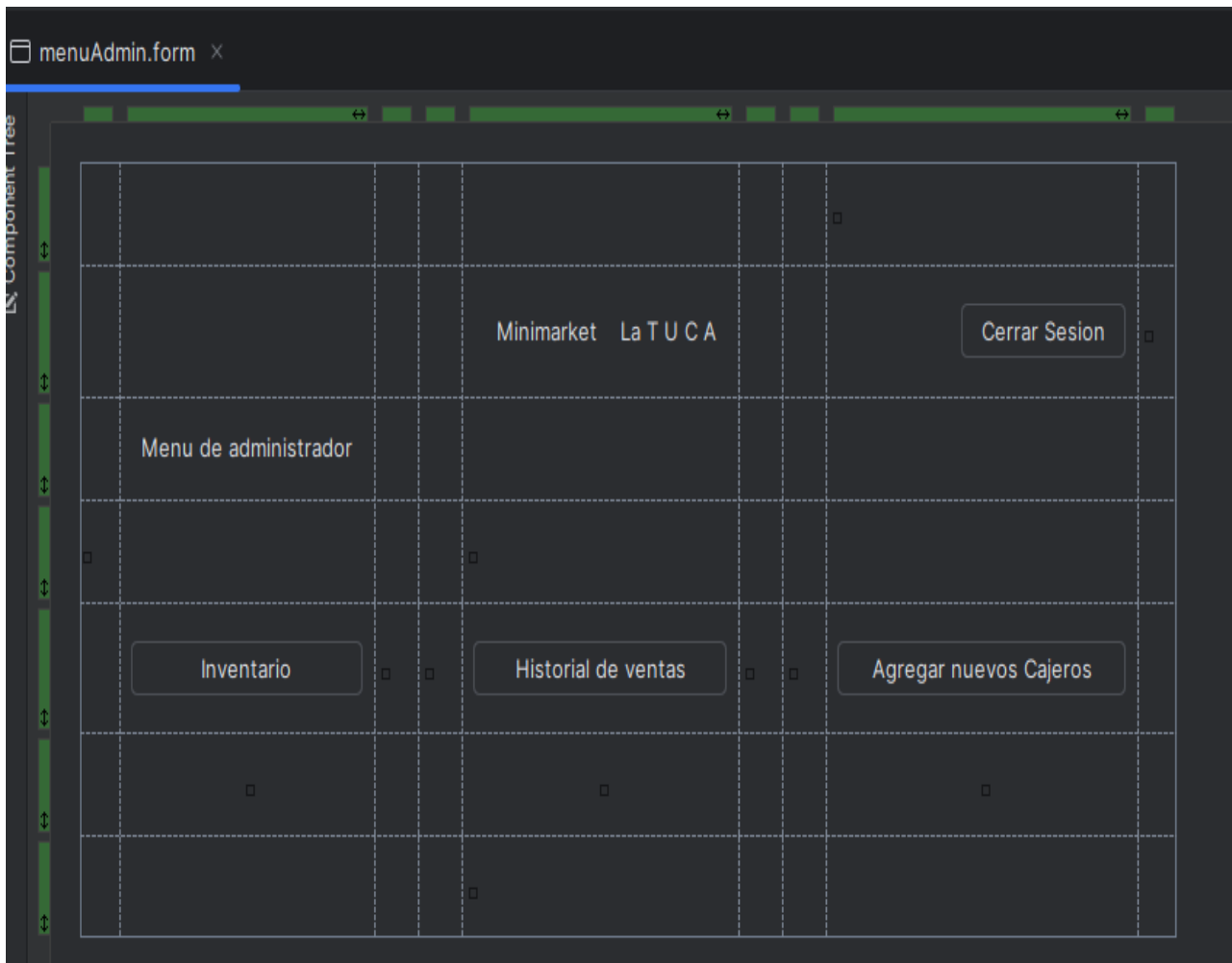
Usuario:

Contraseña:

Modo:

Seleccione un modo

Iniciar Sesión



admin.form inventario.form **AgregarProductos.form** × ActualizarInventario.form

Agregar Productos a minimarket LA T U C A

Codig ID del producto:

Nombre:

Cantidad:

Precio:

Imagen:

admin.form inventario.form AgregarProductos.form **ActualizarInventario.form** ×

Actualizar Stock Productos minimarket La T U C A

Ingresa el codigo ID del producto

Actualizar Cantidad:

Actualizar Precio:

Actualizar Imagen

admin.form

inventario.form

AgregarProductos.form

ActualizarInventario.form

HISTORIAL DE VENTAS

Menu Principal

Ventas Realizadas

ID de la Venta

Fecha de la venta

Total

Recibo

Abrir Recibo

Anterior

Siguiente

AgregarProductos.form

ActualizarInventario.form

historial.form

AgregarCajeros.form

Menu para agregar cajeros nuevos

Regresar

Usuario:

Contraseña:

Agregar Cajero

menuCajeros.form ×

Menu Cajeros

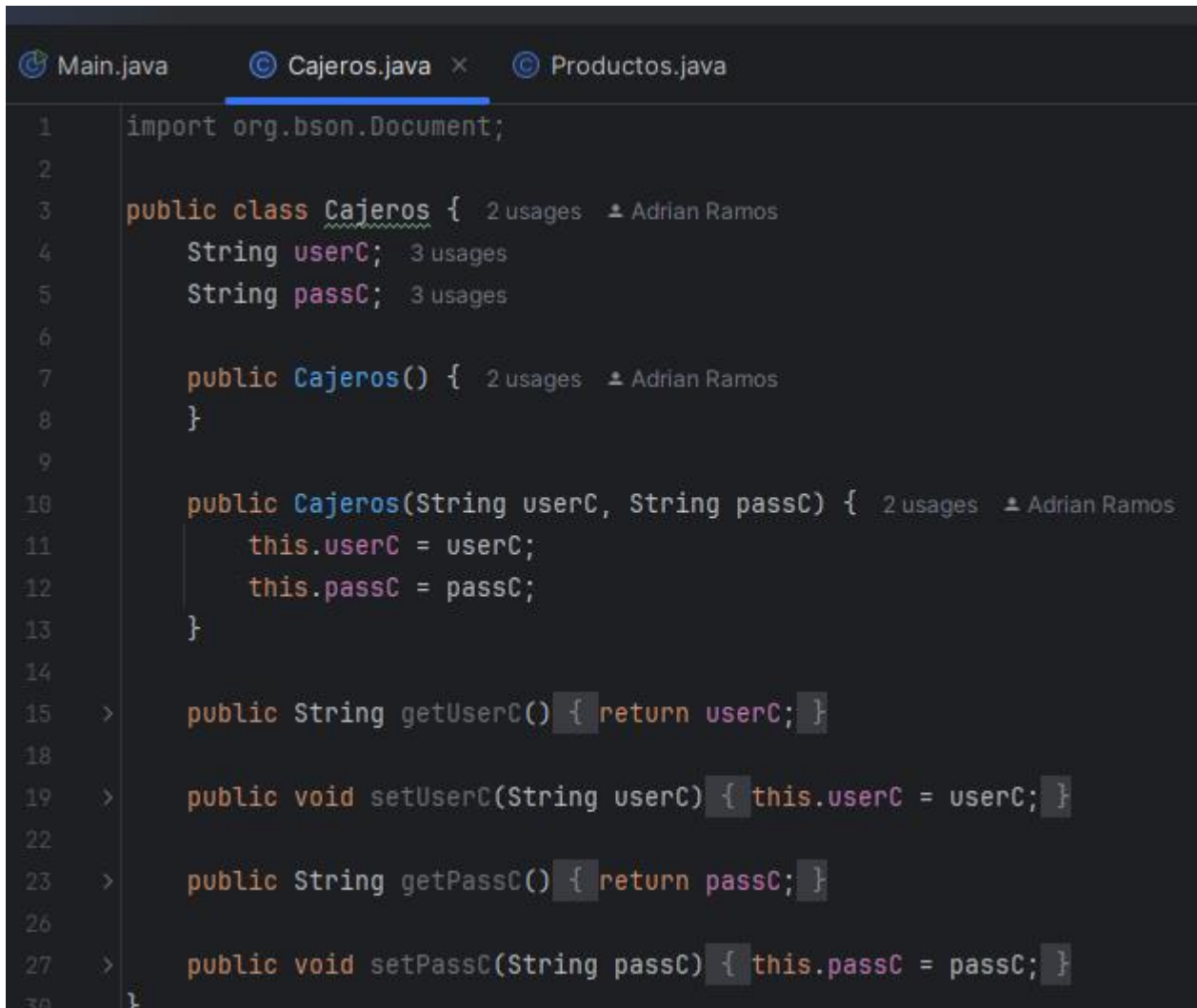
Realizar Transaccion	Finalizar Compra	Total:	Cerrar Sesión
Codigo del producto		Buscar	
Nombre:			
Cantidad:			
Precio:	\$		
	Agregar al carrito		
	Anterior	Siguiente	

DESARROLLO

DOCUMENTACIÓN DE CLASES Y METODOS

Clase Cajero

- Atributos:
 - User
 - Password



```
1  import org.bson.Document;
2
3  public class Cajeros { 2 usages  ± Adrian Ramos
4      String userC; 3 usages
5      String passC; 3 usages
6
7      public Cajeros() { 2 usages  ± Adrian Ramos
8      }
9
10     public Cajeros(String userC, String passC) { 2 usages  ± Adrian Ramos
11         this.userC = userC;
12         this.passC = passC;
13     }
14
15     > public String getUserC() { return userC; }
18
19     > public void setUserC(String userC) { this.userC = userC; }
22
23     > public String getPassC() { return passC; }
26
27     > public void setPassC(String passC) { this.passC = passC; }
30 }
```

Clase Productos

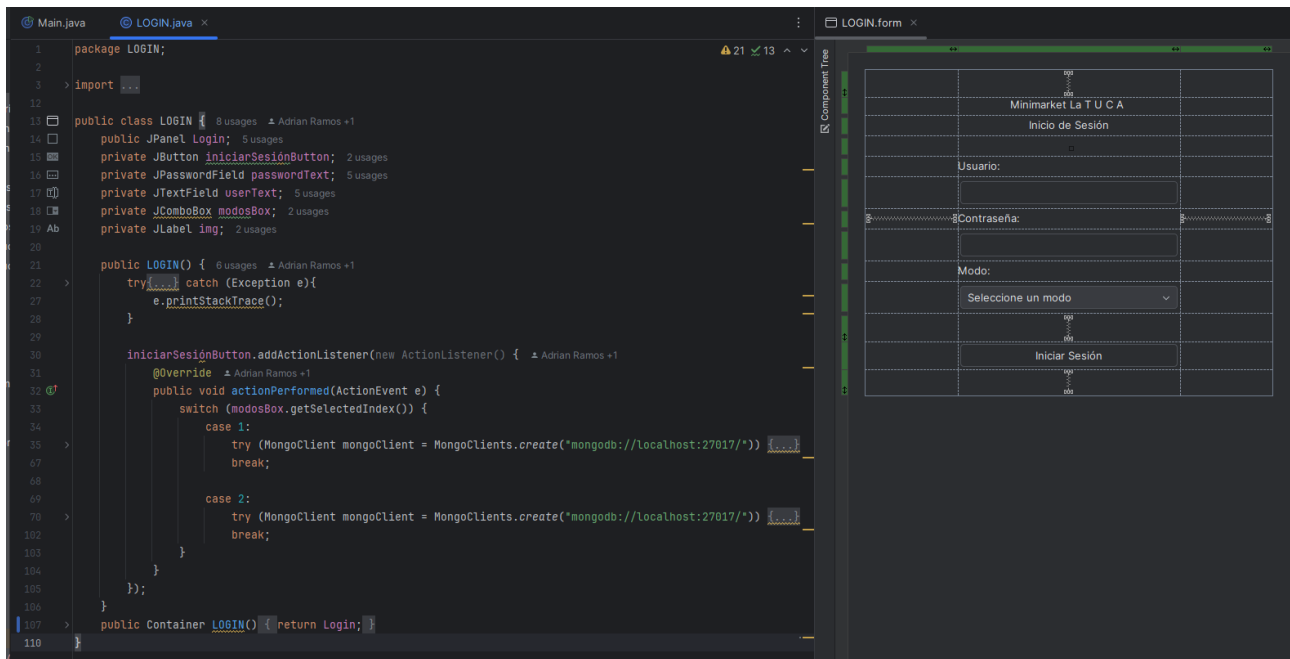
- **Atributos:**
 - **Código**
 - **Nombre**
 - **Precio**
 - **Cantidad**

```
1  import org.bson.Document;
2
3  public class Productos { 5 usages  Luis Adrian Ramos
4      String codigo; 3 usages
5      String nombre; 3 usages
6      double precio; 3 usages
7      int cantidad; 3 usages
8
9      public Productos() { 5 usages  Luis Adrian Ramos
10     }
11
12     public Productos(String codigo, String nombre, double precio, int cantidad) { 5 usages  Luis Adrian Ramos
13         this.codigo = codigo;
14         this.nombre = nombre;
15         this.precio = precio;
16         this.cantidad = cantidad;
17     }
18
19     > public String getCodigo() { return codigo; }
22
23     > public void setCodigo(String codigo) { this.codigo = codigo; }
26
27     > public String getNombre() { return nombre; }
30
31     > public void setNombre(String nombre) { this.nombre = nombre; }
34
35     > public double getPrecio() { return precio; }
38
39     > public void setPrecio(double precio) { this.precio = precio; }
42
43     > public int getCantidad() { return cantidad; }
46
47     > public void setCantidad(int cantidad) { this.cantidad = cantidad; }
50 }
```

Package Login

- Clases:

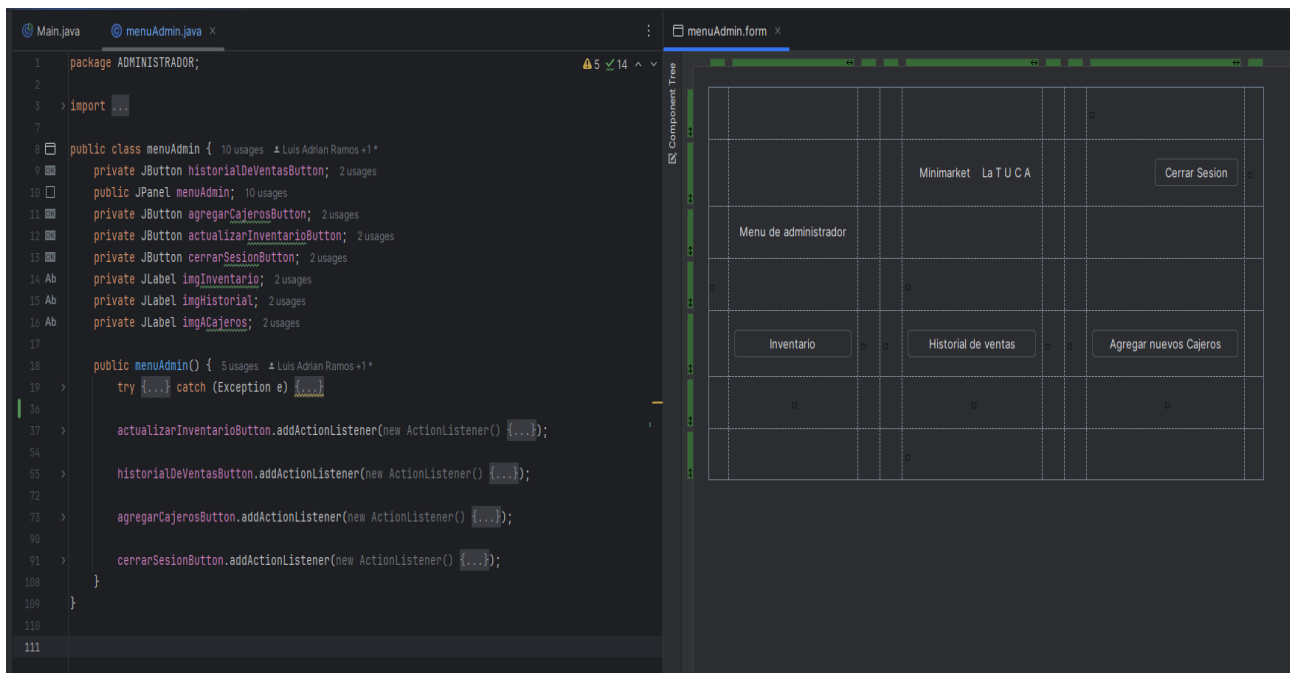
Login



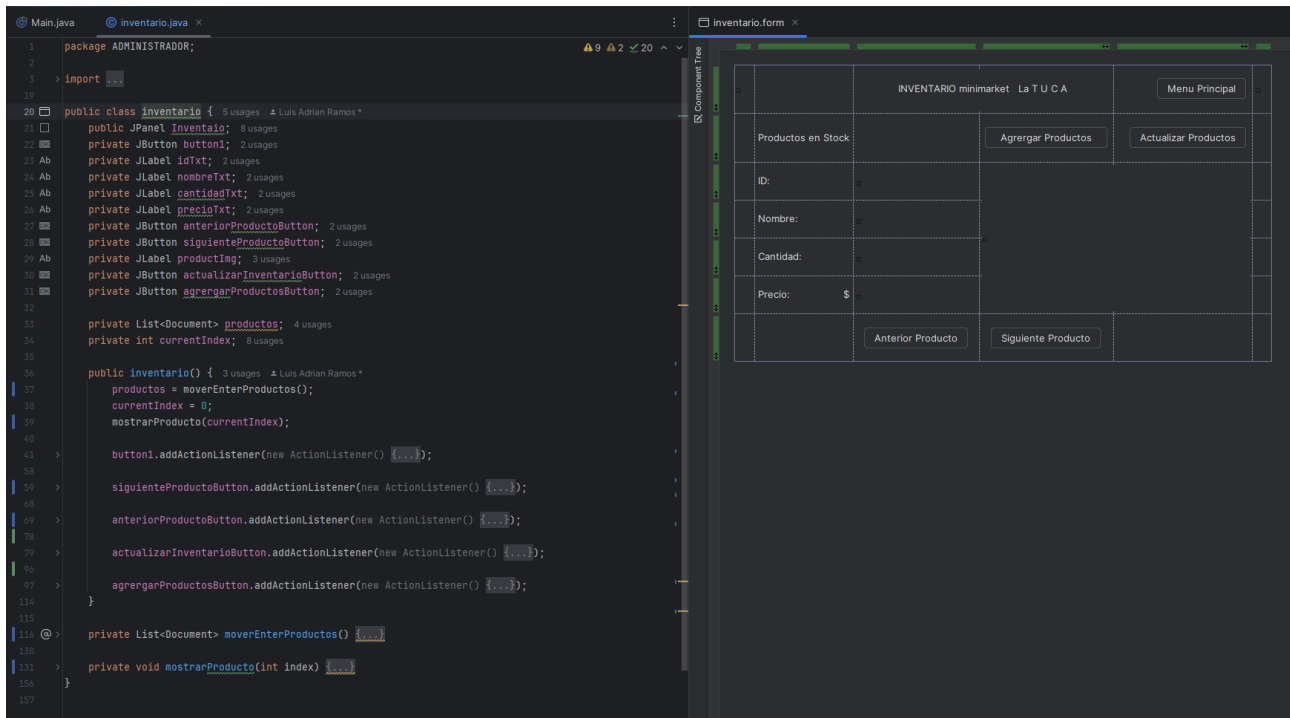
Package Administrador

- Clases:

Menú admin

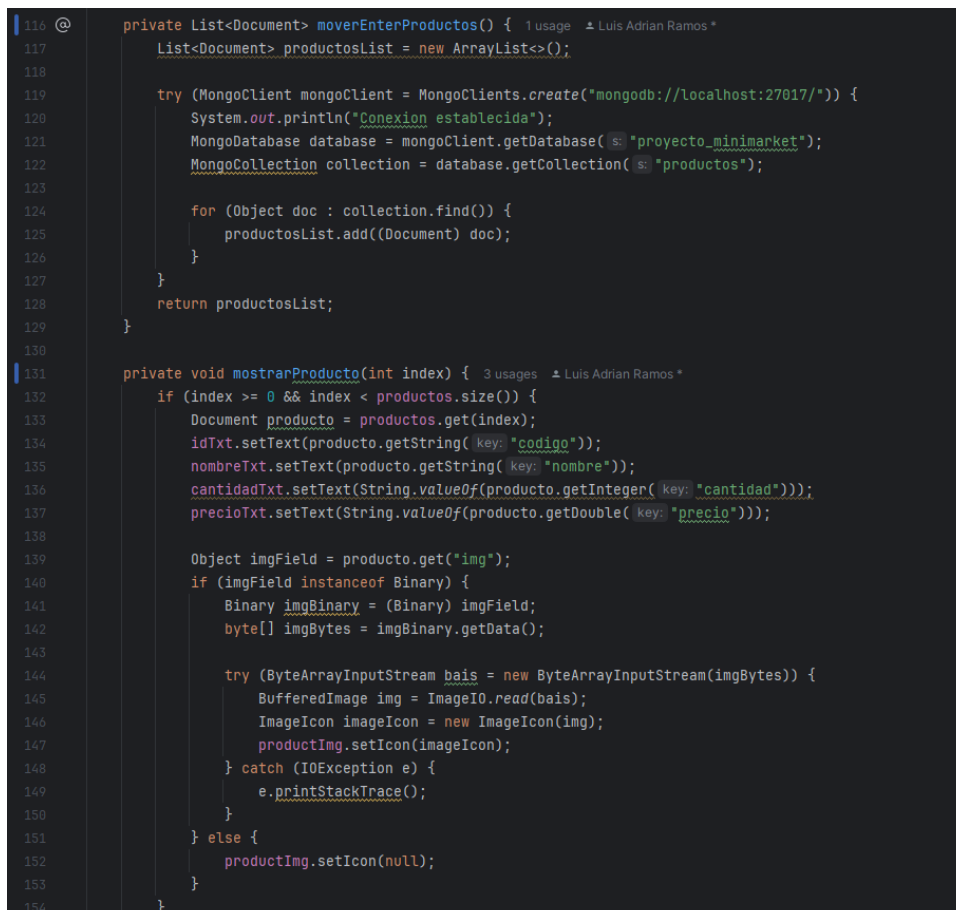


Inventario

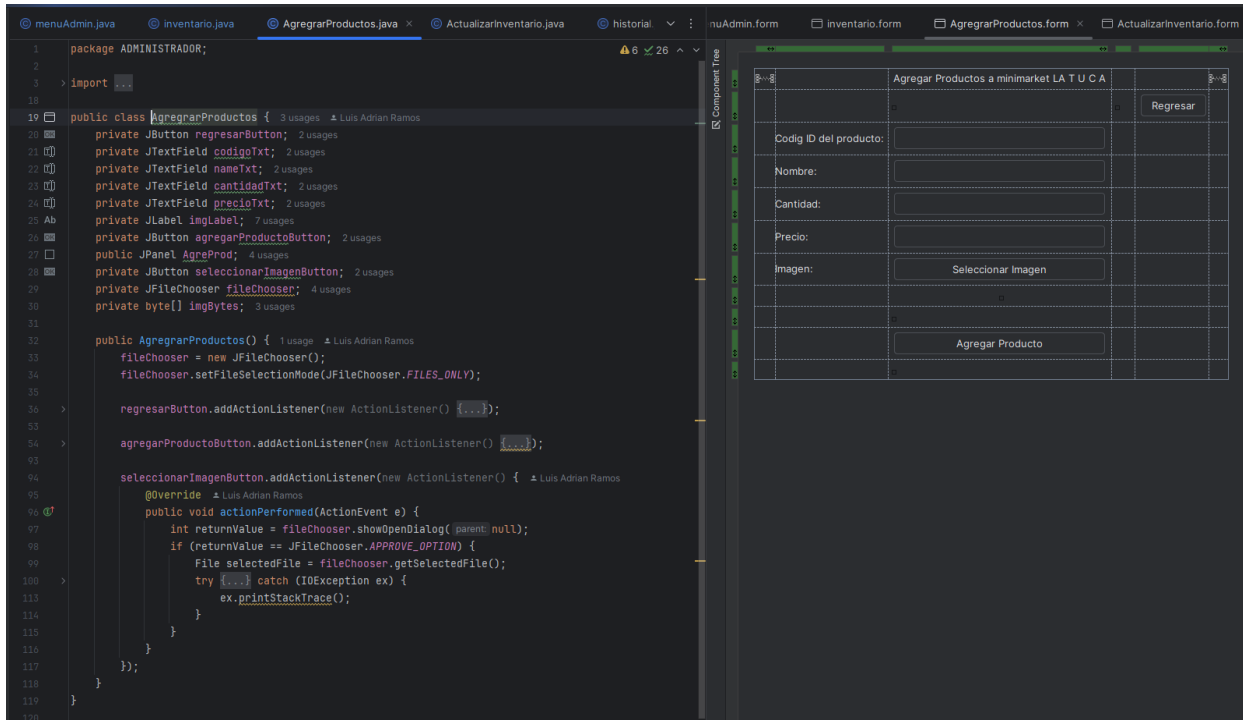


Métodos:

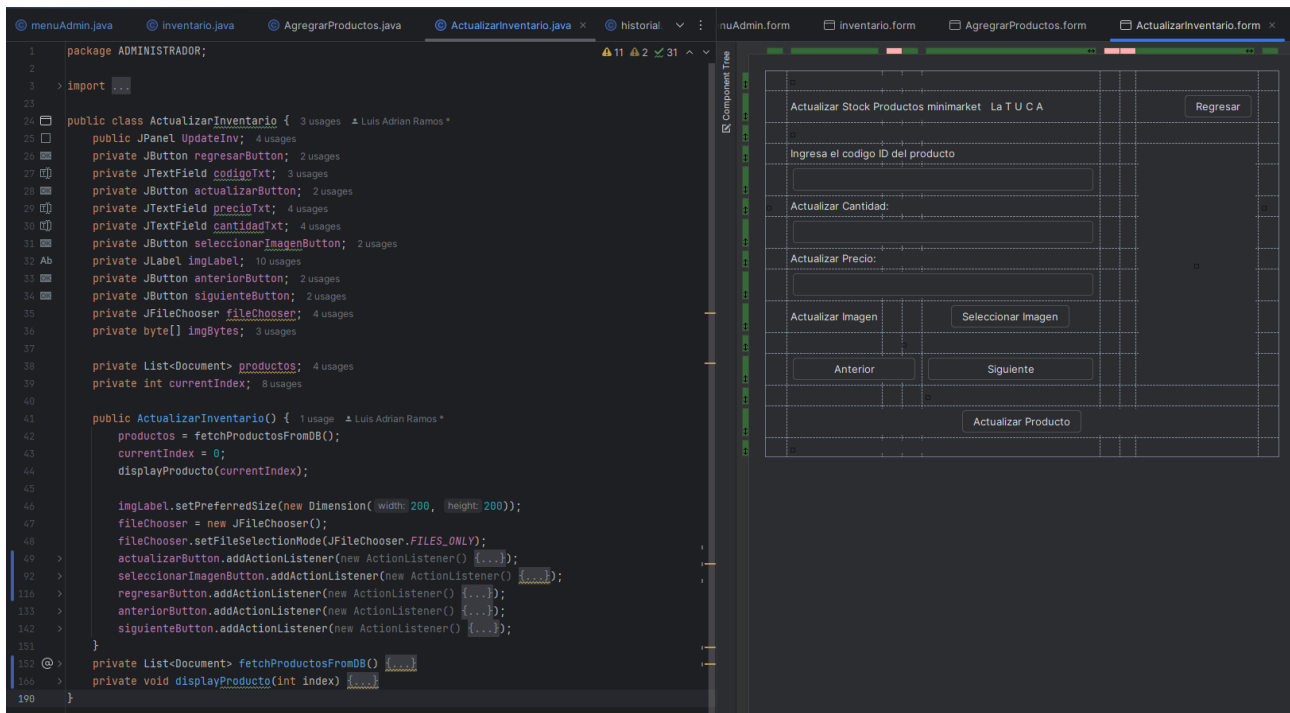
- `moverEntreProductos()`
- `mostrarProductos()`



Agregar Producto



Actualizar Inventario



Métodos:

- **moverEntreProductos()**
- **mostrarProductos()**

```

116 @ private List<Document> moverEnterProductos() { 1 usage Luis Adrian Ramos *
117     List<Document> productosList = new ArrayList<>();
118
119     try (MongoClient mongoClient = MongoClient.create("mongodb://localhost:27017/")) {
120         System.out.println("Conexion establecida");
121         MongoDB database = mongoClient.getDatabase(s: "proyecto_minimarket");
122         MongoCollection collection = database.getCollection(s: "productos");
123
124         for (Object doc : collection.find()) {
125             productosList.add((Document) doc);
126         }
127     }
128     return productosList;
129 }
130
131 private void mostrarProducto(int index) { 3 usages Luis Adrian Ramos *
132     if (index >= 0 && index < productos.size()) {
133         Document producto = productos.get(index);
134         idTxt.setText(producto.getString( key: "codigo"));
135         nombreTxt.setText(producto.getString( key: "nombre"));
136         cantidadTxt.setText(String.valueOf(producto.getInteger( key: "cantidad")));
137         precioTxt.setText(String.valueOf(producto.getDouble( key: "precio")));
138
139         Object imgField = producto.get("img");
140         if (imgField instanceof Binary) {
141             Binary imgBinary = (Binary) imgField;
142             byte[] imgBytes = imgBinary.getData();
143
144             try (ByteArrayInputStream bais = new ByteArrayInputStream(imgBytes)) {
145                 BufferedImage img = ImageIO.read(bais);
146                 ImageIcon imageIcon = new ImageIcon(img);
147                 productImg.setIcon(imageIcon);
148             } catch (IOException e) {
149                 e.printStackTrace();
150             }
151         } else {
152             productImg.setIcon(null);
153         }
154     }

```

Historial

The screenshot displays the 'historial.java' file in an IDE. The code defines a 'historial' class with the following methods:

- `historial()`: Initializes the class, sets the current index to 0, and fetches sales from the database. It also sets up action listeners for the 'Menu Principal', 'Anterior', 'Siguiete', and 'Abrir Recibo' buttons.
- `fetchVentasFromDB()`: A private method to fetch sales from the database.
- `displayRecibo(int index)`: A private method to display a receipt for a specific sale index.

The GUI shows a table with the following columns: 'Ventas Realizadas', 'ID de la Venta', 'Fecha de la venta', and 'Total'. The table contains one row of data. Below the table, there are buttons for 'Anterior', 'Siguiete', and 'Abrir Recibo'. A 'Menu Principal' button is located at the top right of the table area.

Métodos:

- moverEntreVentas()
- mostrarRecibo()

```

103  private List<Document> moverEntreVentas() { 1 usage  Luis Adrian Ramos *
104      List<Document> ventasList = new ArrayList<>();
105
106      try (MongoClient mongoClient = MongoClient.create("mongodb://localhost:27017/")) {
107          MongoDB database = mongoClient.getDatabase("s: "proyecto_minimarket");
108          MongoCollection<Document> collection = database.getCollection("s: "ventas");
109
110          for (Document doc : collection.find()) {
111              ventasList.add(doc);
112          }
113      }
114      return ventasList;
115  }
116
117  private void mostrarRecibo(int index) { 3 usages  Luis Adrian Ramos *
118      if (index >= 0 && index < recibos.size()) {
119          Document recibo = recibos.get(index);
120
121          ventaIDtxt.setText(String.valueOf(recibo.getInteger( key: "ventaID")));
122
123          Object fechaObj = recibo.get("fecha");
124          if (fechaObj instanceof Date) {
125              Date fecha = (Date) fechaObj;
126              SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
127              fechatxt.setText(sdf.format(fecha));
128          } else {
129              fechatxt.setText("Fecha no disponible");
130          }
131
132          totaltxt.setText(String.format("Total: %.2f", recibo.getDouble( key: "total")));
133      }
134  }
135  }

```

Agregar Cajeros

The screenshot displays an IDE with the following components:

- Code Editor (Left):** Contains the Java code for the `AgregarCajeros` class. The code includes package declarations, imports, and methods for handling button clicks and database operations. Key lines include:


```

      package ADMINISTRADOR;

      import ...;

      public class AgregarCajeros {
          public JPanel Acajeros;
          private JButton agregarCajeroButton;
          private JTextField textField1;
          private JTextField textField2;
          private JButton menuPrincipalButton;

          public AgregarCajeros() {
              menuPrincipalButton.addActionListener(new ActionListener() { ... });
              agregarCajeroButton.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                      String username = textField1.getText();
                      String password = textField2.getText();

                      if (username.isEmpty() || password.isEmpty()) { ... }

                      try (MongoClient mongoClient = MongoClient.create("mongodb://localhost:27017/")) {
                          MongoDB database = mongoClient.getDatabase("s: "proyecto_minimarket");
                          MongoCollection<Document> collection = database.getCollection("s: "cajeros");

                          Document query = new Document("user", username);
                          if (collection.find(query).first() != null) { ... }

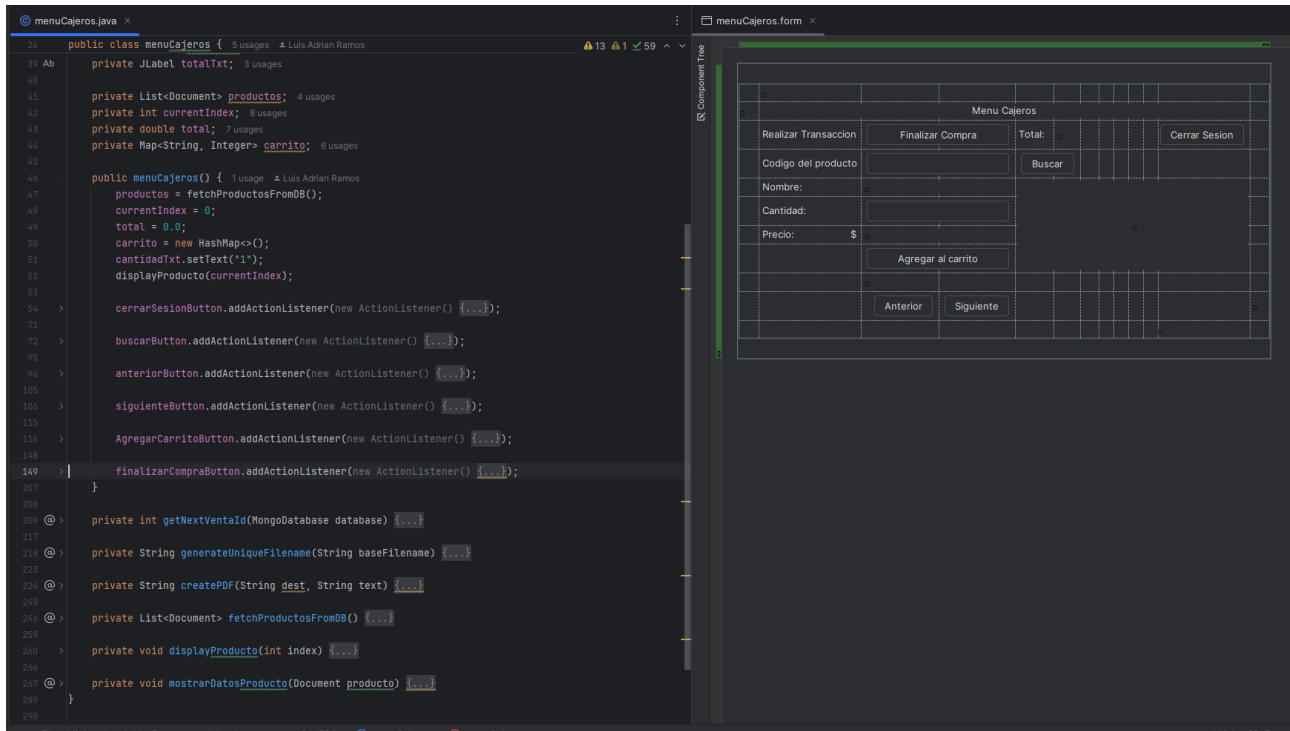
                          Document document = new Document("user", username)
                              .append("password", password);
                          collection.insertOne(document);

                          JOptionPane.showMessageDialog(parentComponent, null, "Cajero agregado exitosamente", ...);
                      } catch (Exception ex) {
                          ex.printStackTrace();
                          JOptionPane.showMessageDialog(parentComponent, null, "Error al agregar el cajero.", ...);
                      }
                  }
              });
          }
      }
      
```
- GUI Preview (Right):** Shows a visual representation of the application window. It features a title bar, a menu bar with "AgregarCajeros", and a main content area with a "Menu para agregar cajeros nuevos" and a "Regresar" button. Below this, there are input fields for "Usuario:" and "Contraseña:", and an "Agregar Cajero" button at the bottom.

Package Cajero

- Clases:

Menú cajero



Métodos:

- contadorVentas()
- nombrePDF()
- crearPDF()

```

209 @
210 private int contadorVentas(MongoDatabase database) {
211     MongoClient<Document> contadorCollection = database.getCollection("$ "contadorVentas");
212     Document query = new Document("$id", "ventaId");
213     Document update = new Document("$inc", new Document("seq", 1));
214     FindOneAndUpdateOptions options = new FindOneAndUpdateOptions().returnDocument(ReturnDocument.AFTER).upsert(true);
215     Document result = contadorCollection.findOneAndUpdate(query, update, options);
216     return result == null ? 1 : result.getInteger("seq");
217 }
218
219 @
220 private String nombrePDF(String baseFilename) {
221     SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
222     String timestamp = sdf.format(new Date());
223     return baseFilename.replace("pdf", "_" + timestamp + ".pdf");
224 }
225
226 @
227 private String crearPDF(String dest, String text) {
228     try {
229         String uniqueDest = nombrePDF(dest);
230
231         com.itextpdf.text.Document document = new com.itextpdf.text.Document();
232         PdfWriter.getInstance(document, new FileOutputStream(uniqueDest));
233
234         document.open();
235
236         Font font = new Font(Font.FontFamily.HELVETICA, 12, Font.NORMAL);
237         document.add(new Paragraph(text, font));
238
239         document.close();
240
241         System.out.println("PDF creado: " + uniqueDest);
242         return uniqueDest;
243     } catch (Exception e) {
244         e.printStackTrace();
245     }
246     return null;
247 }
  
```

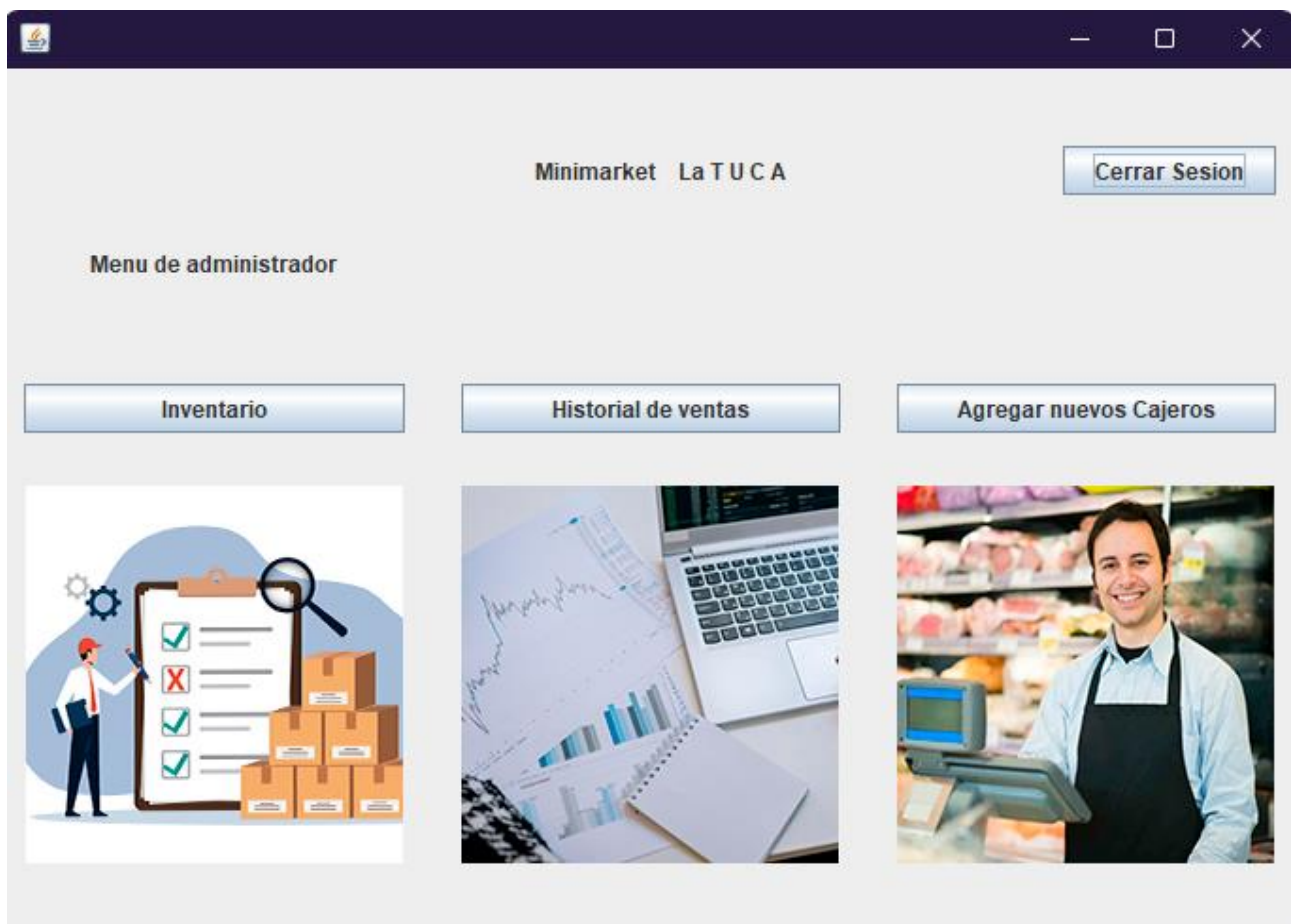
- **moverEntreProductos()**
- **mostrarProductos()**
- **mostrarDatosProductos()**

```
245
246 @ private List<Document> moverEntreProductos() { 1 usage Luis Adrian Ramos *
247     List<Document> productosList = new ArrayList<>();
248
249     try (MongoClient mongoClient = MongoClient.create("mongodb://localhost:27017/")) {
250         MongoDBDatabase database = mongoClient.getDatabase(s: "proyecto_minimarket");
251         MongoCollection<Document> collection = database.getCollection(s: "productos");
252
253         for (Document doc : collection.find()) {
254             productosList.add(doc);
255         }
256     }
257     return productosList;
258 }
259
260 private void mostrarProducto(int index) { 3 usages Luis Adrian Ramos *
261     if (index >= 0 && index < productos.size()) {
262         Document producto = productos.get(index);
263         mostrarDatosProducto(producto);
264     }
265 }
266
267 @ private void mostrarDatosProducto(Document producto) { 2 usages Luis Adrian Ramos
268     codigoTxt.setText(producto.getString(key: "codigo"));
269     nombretxt.setText(producto.getString(key: "nombre"));
270     cantidadTxt.setText("1");
271     preciortxt.setText(String.valueOf(producto.getDouble(key: "precio")));
272
273     Object imgField = producto.get("img");
274     if (imgField instanceof Binary) {
275         Binary imgBinary = (Binary) imgField;
276         byte[] imgBytes = imgBinary.getData();
277
278         try (ByteArrayInputStream bais = new ByteArrayInputStream(imgBytes)) {...} catch (IOException e) {
279             e.printStackTrace();
280         }
281     } else {
282         img.setIcon(null);
283     }
284 }
285
286
287
288 }
```

DESPLIEGUE



A screenshot of a web application window titled "Login". The window has a dark purple header bar with the title and standard window controls. The main content area is light gray and contains the text "Minimarket La T U C A" and "Inicio de Sesión" centered at the top. Below this is a large circular icon representing a user. Underneath the icon are three input fields: "Usuario:" (a text box), "Contraseña:" (a password box), and "Modo:" (a dropdown menu with the text "Seleccione un modo"). At the bottom of the form is a blue button labeled "Iniciar Sesión".



Inventario

— □ ×

INVENTARIO minimarket La T U C A

Menu Principal

Productos en Stock

Agregar Productos


Actualizar Productos

ID: 0001

Nombre: papas

Cantidad: 39

Precio: \$ 0.5



Anterior Producto

Siguiente Producto

Agregar Productos

— □ ×

Agregar Productos a minimarket LA T U C A

Regresar

Codig ID del producto:

Nombre:

Cantidad:

Precio:

Imagen:

Seleccionar Imagen

Agregar Producto

Actualizar Inventario

Actualizar Stock Productos minimarket La T U C A

Regresar

Ingresa el codigo ID del producto

0001

Actualizar Cantidad:


39

Actualizar Precio:

0.5

Actualizar Imagen

Seleccionar Imagen



Anterior

Siguiente

Actualizar Producto

Historial de Ventas

HISTORIAL DE VENTAS

Menu Principal

Ventas Realizadas

ID de la Venta 1

Fecha de la venta 2024-08-05 20:35:09

Total Total: 2,00

Recibo

Abrir Recibo

Anterior

Siguiente

Agregar Cajeros

Menu para agregar cajeros nuevos

Regresar

Usuario:

Contraseña:

Agregar Cajero

Menu Cajeros

Realizar Transaccion

Finalizar Compra

Total:

Cerrar Sesion

Codigo del producto

0001

Buscar

Nombre:

papas

Cantidad:

1

Precio:

\$

0.5

Agregar al carrito

Anterior

Siguiente

