**Documentacion JDOK:**

**Producto Controlador:**

**package main.java.org.example.sistemaproyec.Controlador;**

**import main.java.org.example.sistemaproyec.Modelo.Producto;**

**import java.util.List;**

**import java.util.Scanner;**

**/\*\***

**\* Controlador encargado de gestionar las operaciones relacionadas con los productos.**

**\* Permite agregar, editar y eliminar productos del sistema.**

**\*/**

**public class ProductoControlador {**

**private List<Producto> productos;**

**private Scanner scanner = new Scanner(System.in);**

**/\*\***

**\* Agrega un nuevo producto al sistema. Solicita al usuario los datos del producto y**

**\* los almacena en la lista de productos.**

**\*/**

**public void agregarProducto() {**

**// ... (implementación con validación y uso de List)**

**}**

**/\*\***

**\* Edita un producto existente. Permite al usuario seleccionar el producto a editar**

**\* y modificar los campos deseados.**

**\***

**\* @param indice El índice del producto a editar en la lista.**

**\*/**

**public void editarProducto(int indice) {**

**// ... (implementación con validación y manejo de índices fuera de rango)**

**}**

**/\*\***

**\* Elimina un producto del sistema.**

**\***

**\* @param indice El índice del producto a eliminar en la lista.**

**\*/**

**public void eliminarProducto(int indice) {**

**// ... (implementación con manejo de índices fuera de rango)**

**}**

**}**

**MainApp**

**Java**

**package main.java.org.example.sistemaproyec;**

**import javafx.application.Application;**

**import javafx.fxml.FXMLLoader;**

**import javafx.scene.Scene;**

**import javafx.stage.Stage;**

**/\*\***

**\* Clase principal de la aplicación de gestión de productos.**

**\* Extiende la clase `Application` de JavaFX y se encarga de iniciar la interfaz gráfica**

**\* de usuario (GUI) cargando el archivo FXML correspondiente a la vista de productos.**

**\***

**\* @author (Tu nombre aquí) // Opcional, agrega tu nombre como autor**

**\*/**

**public class MainApp extends Application {**

**/\*\***

**\* Método principal de la aplicación invocado por JavaFX.**

**\* Se encarga de la configuración inicial de la ventana principal (Stage) y la carga de la interfaz gráfica.**

**\***

**\* @param primaryStage La ventana principal de la aplicación.**

**\* @throws Exception Si ocurre un error al cargar el archivo FXML.**

**\*/**

**@Override**

**public void start(Stage primaryStage) throws Exception {**

**// Carga el archivo FXML de la vista de productos**

**FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/main/resources/org/example/sistemaproyec/Vista/ProductoVista.fxml"));**

**Scene scene = new Scene(fxmlLoader.load(), 400, 300); // Define el tamaño inicial de la ventana**

**// Configura la ventana principal**

**primaryStage.setTitle("Sistema de Gestión de Materiales - Productos");**

**primaryStage.setScene(scene);**

**primaryStage.show();**

**}**

**/\*\***

**\* Método de punto de entrada de la aplicación.**

**\* Lanza la aplicación JavaFX.**

**\***

**\* @param args Argumentos de línea de comando (no utilizados en este caso).**

**\*/**

**public static void main(String[] args) {**

**launch(args);**

**}**

**}**

**Application**

**package javafx.application;**

**/\*\***

**\* Clase abstracta base para las aplicaciones JavaFX.**

**\***

**\* Las clases que extienden `Application` deben implementar el método `start(Stage stage)`**

**\* para definir la interfaz gráfica de usuario (GUI) de la aplicación.**

**\*/**

**public abstract class Application {**

**/\*\***

**\* Constante que define el nombre del tema Caspian para JavaFX.**

**\*/**

**public static final String STYLESHEET\_CASPIAN = "CASPIAN";**

**/\*\***

**\* Constante que define el nombre del tema Modena para JavaFX.**

**\*/**

**public static final String STYLESHEET\_MODENA = "MODENA";**

**/\*\***

**\* Objeto que proporciona acceso a servicios del sistema operativo relacionados con la aplicación.**

**\* (La implementación no está disponible).**

**\*/**

**private javafx.application.HostServices hostServices;**

**/\*\***

**\* Almacena la hoja de estilo predeterminada basada en el agente de usuario.**

**\* (La implementación no está disponible).**

**\*/**

**private static java.lang.String userAgentStylesheet;**

**/\*\***

**\* Lanza la aplicación JavaFX.**

**\***

**\* @param aClass La clase que extiende `Application` que define la aplicación.**

**\* @param strings Argumentos de línea de comando (opcional).**

**\*/**

**public static void launch(Class<? extends Application> aClass, String... strings) {**

**// Implementación no disponible (código compilado)**

**}**

**/\*\***

**\* Lanza la aplicación JavaFX.**

**\***

**\* @param strings Argumentos de línea de comando (opcional).**

**\*/**

**public static void launch(String... strings) {**

**// Implementación no disponible (código compilado)**

**}**

**/\*\***

**\* Constructor por defecto.**

**\* (La implementación no está disponible).**

**\*/**

**public Application() {**

**// Implementación no disponible (código compilado)**

**}**

**/\*\***

**\* Permite a la aplicación realizar inicializaciones previas al método `start`.**

**\* Se puede utilizar para cargar recursos o realizar tareas de configuración.**

**\***

**\* @throws Exception Si ocurre un error durante la inicialización.**

**\*/**

**public void init() throws Exception {**

**// Implementación no disponible (código compilado)**

**}**

**/\*\***

**\* Método abstracto que define la interfaz gráfica de usuario (GUI) de la aplicación.**

**\* Este método es invocado por JavaFX para inicializar la ventana principal y cargar**

**\* el contenido de la aplicación.**

**\***

**\* @param stage La ventana principal de la aplicación.**

**\* @throws Exception Si ocurre un error durante la creación de la GUI.**

**\*/**

**public abstract void start(Stage stage) throws Exception;**

**/\*\***

**\* Permite a la aplicación realizar tareas de limpieza y cierre al finalizar su ejecución.**

**\* Se puede utilizar para liberar recursos o guardar datos.**

**\***

**\* @throws Exception Si ocurre un error durante el cierre de la aplicación.**

**\*/**

**public void stop() throws Exception {**

**// Implementación no disponible (código compilado)**

**}**

**/\*\***

**\* Proporciona acceso al objeto `HostServices` que permite acceder a servicios del sistema operativo.**

**\* (La implementación no está disponible).**

**\***

**\* @return Objeto `HostServices` asociado a la aplicación.**

**\*/**

**public final javafx.application.HostServices getHostServices() {**

**// Implementación no disponible (código compilado)**

**}**

**/\*\***

**\* Proporciona acceso a los parámetros de la aplicación pasados a través de la línea de comando.**

**\* (La implementación no está disponible).**

**\***

**\* @return Objeto `Parameters` que encapsula los parámetros de la aplicación.**

**\*/**

**public final Application.Parameters getParameters() {**

**// Implementación no disponible (código compificado)**

**}**

**/\*\***

**\* Notifica a un precargador (opcional) sobre el estado de la aplicación.**

**\* (La implementación no está disponible).**

**\***

**\* @param preloaderNotification Notificación a enviar al precargador.**

**\*/**

**public final void notifyPreloader(Preloader.PreloaderNotification preloaderNotification) {**

**// Implementación no disponible**

**ProductoVista.Java:**

**package main.resources.org.example.sistemaproyec.Vista;**

**import javafx.fxml.FXML;**

**import javafx.scene.control.TextField;**

**import main.java.org.example.sistemaproyec.Controlador.ProductoControlador;**

**/\*\***

**\* Controlador FXML para la vista de productos.**

**\***

**\* Esta clase se encarga de interactuar con los elementos gráficos de la vista**

**\* relacionada con la gestión de productos y delega la lógica de negocio al**

**\* `ProductoControlador`.**

**\*/**

**public class ProductoVista {**

**private ProductoControlador controlador = new ProductoControlador();**

**/\*\***

**\* Campo de texto para ingresar el nombre del producto.**

**\*/**

**@FXML**

**private TextField nombreTextField;**

**/\*\***

**\* Campo de texto para ingresar la descripción del producto.**

**\*/**

**@FXML**

**private TextField descripcionTextField;**

**/\*\***

**\* Campo de texto para ingresar el precio del producto.**

**\*/**

**@FXML**

**private TextField precioTextField;**

**/\*\***

**\* Campo de texto para ingresar la cantidad disponible del producto.**

**\*/**

**@FXML**

**private TextField cantidadTextField;**

**/\*\***

**\* Manejador del evento "Agregar Producto".**

**\***

**\* Se encarga de obtener los valores de los campos de texto y llamar al método**

**\* correspondiente en el `ProductoControlador` para agregar un nuevo producto.**

**\* (La implementación específica del método depende del controlador).**

**\*/**

**@FXML**

**protected void agregarProducto() {**

**// Obtener valores de los campos de texto y agregar un producto**

**}**

**/\*\***

**\* Manejador del evento "Editar Producto".**

**\***

**\* Se encarga de invocar la funcionalidad para editar un producto existente.**

**\* (La implementación específica del método depende del controlador).**

**\*/**

**@FXML**

**protected void editarProducto() {**

**// Lógica para editar producto**

**}**

**/\*\***

**\* Manejador del evento "Eliminar Producto".**

**\***

**\* Se encarga de invocar la funcionalidad para eliminar un producto existente.**

**\* (La implementación específica del método depende del controlador).**

**\*/**

**@FXML**

**protected void eliminarProducto() {**

**// Lógica para eliminar producto**

**}**

**}**

**ProductoVista.fxml:**

**Elementos:**

* **VBox: Contenedor principal con alineación central, espaciado de 10 pixeles y fondo gris (#808080).**
* **padding: Margen interior de 10 pixeles en todos los lados.**
* **ImageView: Imagen con un tamaño fijo (ancho 200px, alto 80px) ubicada en la parte superior. La ruta a la imagen se encuentra en Downloads/Diseño\_sin\_título\_\_7\_-removebg-preview.png (se recomienda actualizar esta ruta si cambia la ubicación de la imagen).**
* **TextField (x3): Campos de texto para ingresar datos del producto:** 
  + **nombreTextField: Nombre del producto (con fuente Arial Narrow, tamaño 14px).**
  + **descripcionTextField: Descripción del producto (con fuente Arial Narrow, tamaño 14px).**
  + **precioTextField: Precio del producto (con fuente Arial Narrow, tamaño 14px).**
* **TextField: Campo de texto para ingresar la cantidad disponible del producto (con fuente Arial Narrow, tamaño 14px).**
* **Button (x3): Botones para interactuar con los productos:** 
  + **agregarProducto: Botón para agregar un nuevo producto (invoca el método agregarProducto del controlador).**
  + **editarProducto: Botón para editar un producto existente (invoca el método editarProducto del controlador).**
  + **eliminarProducto: Botón para eliminar un producto existente (invoca el método eliminarProducto del controlador).**