Pontifica Universidad Javeriana

Diagramas

Fundamentos de Ingeniería de Software

Laura Sofia Aponte Sánchez

Juan Esteban Bello Durango

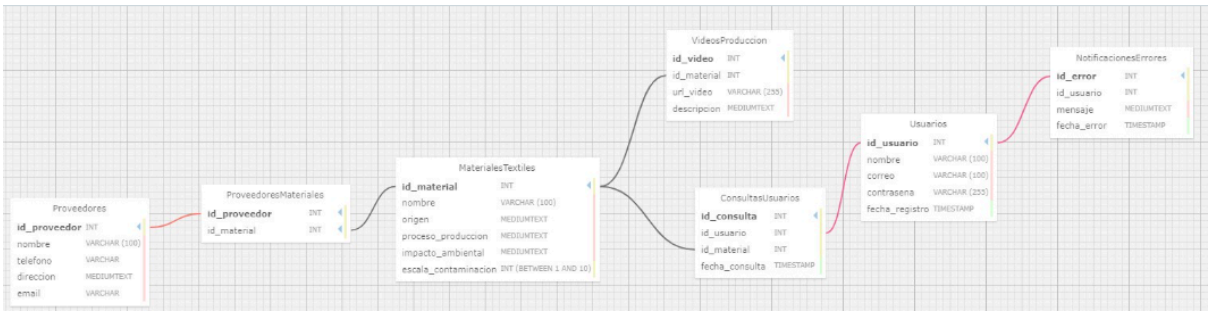
Santiago Galindo Cubillos

Juan Felipe Gutiérrez

Luis Gabriel Moreno Sandoval

Bogotá D.C

Diagrama base de datos



Este diagrama representa el **modelo entidad-relación** (MER) de una base de datos enfocada en materiales textiles, sus proveedores, el impacto ambiental de su producción, y la interacción de usuarios con dicha información. A continuación te explico cada parte del modelo:

Entidades y relaciones principales:

1. **Proveedores**
    - `id_proveedor`: identificador del proveedor.
    - `nombre`, `telefono`, `direccion`, `email`: datos de contacto.
    - Relacionado con los materiales mediante la tabla intermedia **ProveedoresMateriales** (relación muchos a muchos).
2. **MaterialesTextiles**
    - `id_material`: identificador del material textil.
    - `nombre`, `origen`, `proceso_produccion`, `impacto_ambiental`.
    - `escala_contaminacion`: nivel del 1 al 10.
    - Está relacionado con:
        - **Proveedores** a través de **ProveedoresMateriales**.
        - **VideosProduccion** (relación uno a muchos).
        - **ConsultasUsuarios** (para registrar qué materiales han consultado los usuarios).

3. **VideosProduccion**
    - Contiene videos asociados a un material textil.
    - `url_video`, `descripcion`, y la relación con `id_material`.
4. **Usuarios**
    - `id_usuario`, `nombre`, `correo`, `contraseña`, `fecha_registro`.
    - Relacionado con:
        - **ConsultasUsuarios**: materiales que consulta cada usuario.
        - **NotificacionesErrores**: para registrar errores en el sistema asociados a usuarios.
5. **ConsultasUsuarios**
    - Tabla que registra qué usuario consultó qué material y cuándo.
6. **NotificacionesErrores**
    - Registra errores del sistema asociados a un usuario.
    - `mensaje` describe el error y `fecha_error` cuándo ocurrió.

¿Qué se puede hacer con esta base?

- Ver qué proveedores ofrecen un material específico.
- Conocer el impacto ambiental de cada material.
- Consultar los procesos de producción y ver videos explicativos.
- Rastrear el uso de la plataforma por parte de los usuarios.
- Identificar errores y mejorar la experiencia del usuario.

Historia de Usuario: El usuario debe poder registrarse.

Diagrama de clases:

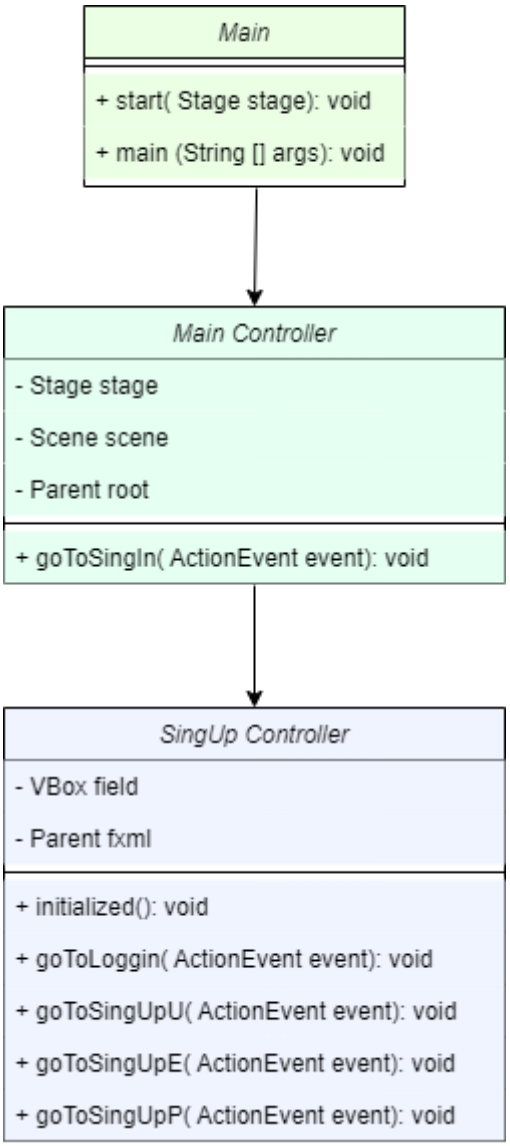
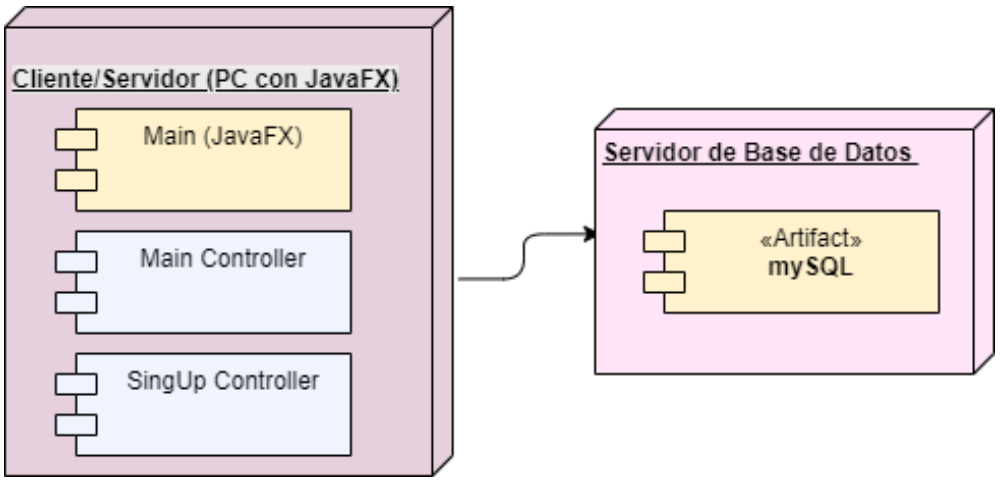


Diagrama de despliegue:



La historia de usuario que aparece en el documento **“El usuario debe poder registrarse”** representa una funcionalidad esencial del sistema desde el punto de vista del usuario final. Aquí te explico brevemente lo que significa y cómo se relaciona con el desarrollo de software:

🔹 Historia de Usuario

"El usuario debe poder registrarse."

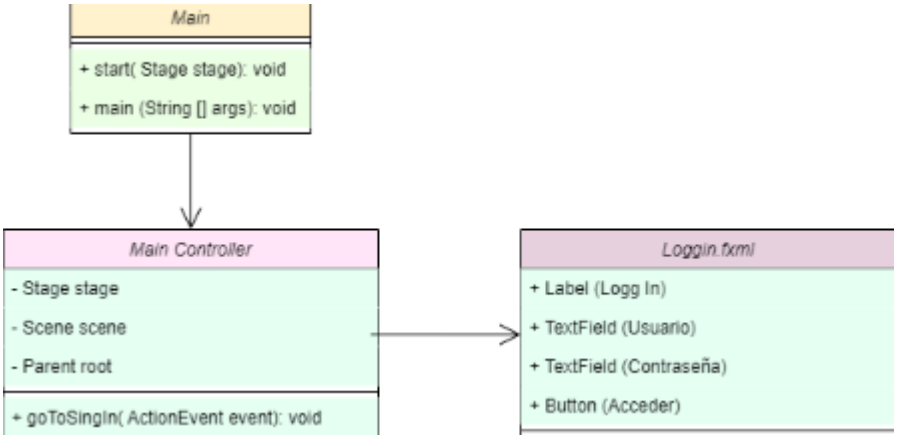
📌 ¿Qué representa?

Es una **historia de usuario** típica en metodologías ágiles como Scrum, que describe una funcionalidad mínima pero valiosa del sistema. En este caso:

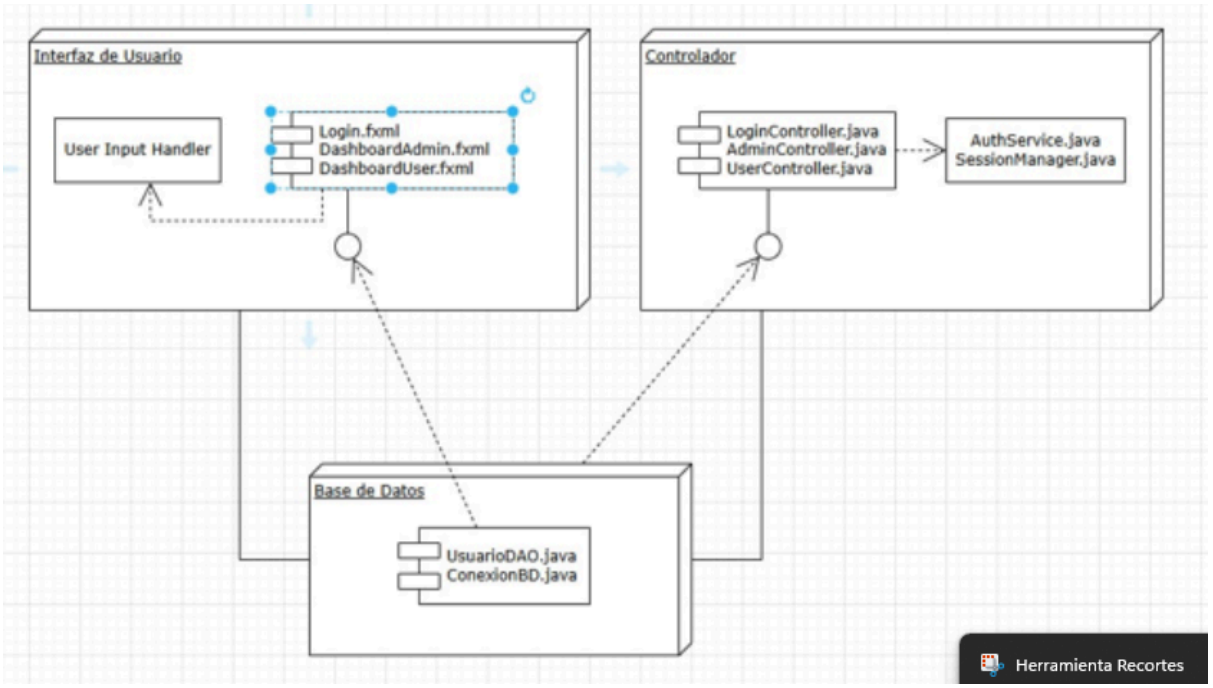
- **Quién:** el usuario (persona que usará la plataforma).
- **Qué quiere hacer:** registrarse en el sistema.
- **Para qué:** para poder usar las funcionalidades protegidas o personalizadas del sistema (como consultar materiales, recibir notificaciones, etc.).

Historia de usuario: el usuario desea iniciar sesion

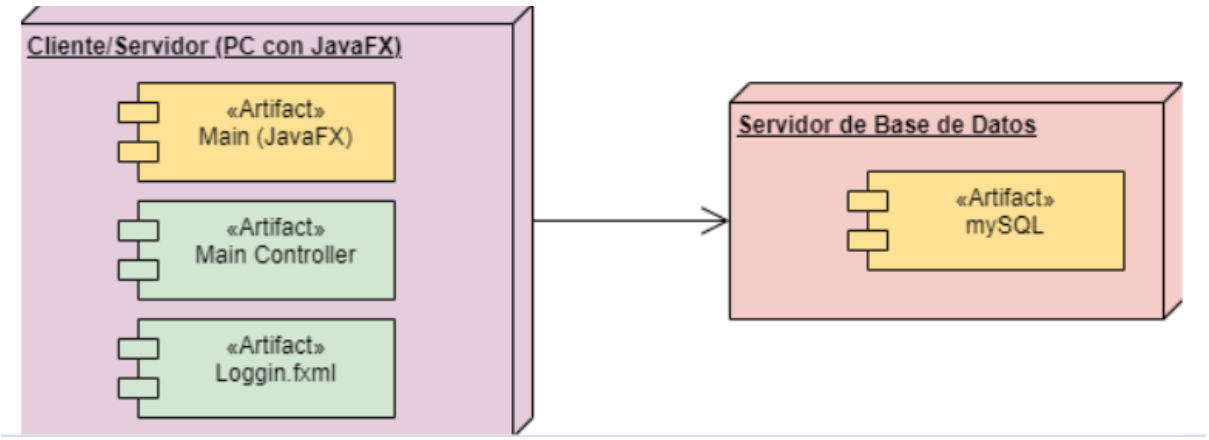
clases



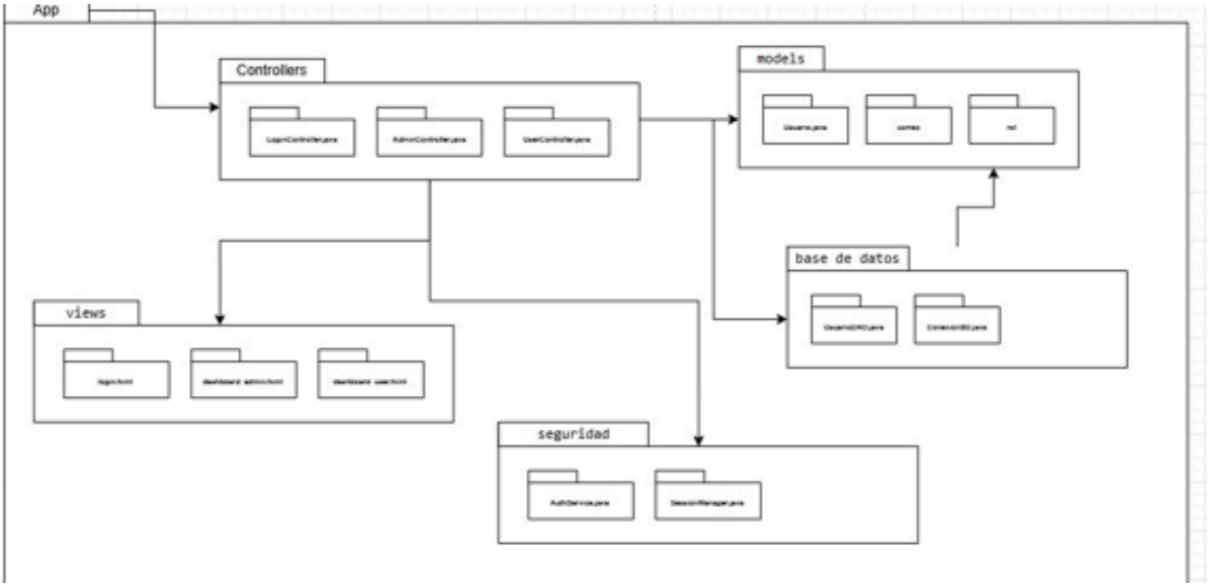
componentes



despliegue



paquetes



Descripcion

1. Diagrama de Clases (primera imagen)

Este diagrama representa las clases Java y sus relaciones dentro del módulo de login.

Main: Clase que arranca la aplicación JavaFX con start() y main().

Main Controller: Controlador principal que maneja la vista Login.fxml. Contiene referencias a Stage, Scene, y un método goToSingIn() que probablemente cambia la escena.

Login.fxml: Archivo de diseño en FXML que incluye:

Label (Logg In)

TextField (Usuario)

TextField (Contraseña)

Button (Acceder)

👉 Este diagrama se enfoca en estructura y funcionalidad de clases relacionadas al login.

2. Diagrama de Componentes (segunda imagen)
Muestra cómo se organizan y conectan los componentes del sistema:

Interfaz de Usuario: Incluye archivos FXML como Login.fxml, DashboardAdmin.fxml, y DashboardUser.fxml.

Controlador: Contiene clases que manejan la lógica de las vistas (LoginController.java, AdminController.java, UserController.java).

Base de Datos: Accedida mediante UsuarioDAO.java y ConexionBD.java.

Servicios de Autenticación: AuthService.java y SessionManager.java permiten login y manejo de sesiones.

👉 Este diagrama es útil para entender cómo se comunica cada módulo de manera estructural.

3. Diagrama de Despliegue (tercera imagen)
Muestra cómo se distribuyen los componentes en el entorno:

Cliente/Servidor (PC con JavaFX):

Main (JavaFX): arranque de la aplicación.

Main Controller: lógica de la interfaz.

Login.fxml: vista de la interfaz.

Servidor de Base de Datos: Componente externo que representa a MySQL.

👉 Este diagrama representa dónde vive cada parte del sistema (cliente vs servidor).

4. Diagrama de Paquetes (cuarta imagen)
Organiza el proyecto en paquetes lógicos:

Controllers: Controladores como LoginController.java, AdminController.java, etc.

Models: Clases que representan entidades como Usuario.java.

Views: Interfaces visuales Login.fxml, DashboardAdmin.fxml, etc.

Seguridad: AuthService.java, SessionManager.java.

Base de Datos: Clases DAO y conexión (UsuarioDAO.java, ConexionBD.java).

👉 Este diagrama muestra la arquitectura modular del sistema, ideal para mantener la organización y escalabilidad.