

**CENTRO EDUCATIVO TECNICO LABORAL KINAL**

5TO. PERITO EN INFORMATICA

Grupo 1

Entrega Manual Final 5TO BIM

Integrantes del Grupo:

Giovanni Emmanuel Carrera Garrido :

Adrian Alessandro Arbizú del Cid:

Esteban Adolfo Cano Gonzales :

Luis Javier Pérez Monzón:

Fernando David Choc Baltazar:

Guatemala, 27/09/2024.

La aplicación que elegimos como grupo fue la de una agencia de viajes lo cual las entidades que lleva son:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Viajero Id (Cliente) | RutaId | VehiculoId | ParadaId | OfertasId | ComentarioId | Itinerario |
| Nombre | Nombre | Marca | Nombre | Fechallegada | Calificacion | RutaId |
| Apellido | Distancia | Modelo | Ubicación | FechaSalida | Comentario | ParadaId |
| Correo | Duración | TipoVehiculo | Tiempo | Preciopasajero | Fecha | Fecha |
| Teléfono |  | Capacidad | Tipoparada | VehiculoId | ViajeroId | Hora |
| Fecha Registró |  |  |  | RutaId | OfertaId |  |
| Itinerario Id |  |  |  |  |  |  |

* Como funciona el proyecto y como lo podemos ejecutar, este proyecto esta construido por un Backend y un Frontend

Backend: Utilizamos la tecnología de Spring Boot para manejar la parte lógica de proyecto, interacción del manejo de datos que se le mandan al proyecto sobre la base de datos y Api Rest para que se pueda comunicar con nuestro frontend.

Frontend: Parte Grafica de nuestro proyecto, lo cual se utilizo JavaFXML para la interfaz del usuario, donde se crearon y se manejaran vistas para la interacción directa del usuario

* Descripción de los componentes principales del proyecto:

Backend (Sping Boot)

* Controller (Controladores) Estos controladores son parte para manejar las solicitudes HTTP y conectar la lógica al frontend. Los controladores usan Anotaciones de Spring como @RestController y @RequestMapping para definir las rutas de la API.

Por ejemplo:

OfertaController: Controlador REST que maneja las operaciones relacionadas con las ofertas, como la creación, edición, eliminación, y obtención de datos de las ofertas de vehículos.

Service (Service) Son clases donde se guarda la lógica, los servicios se encargan de realizar operaciones sobre los datos enviados, como validaciones antes de que lleguen a la base de datos y a los controladores.

Un ejemplo seria:

OfertaService: Proporciona métodos para buscar, editar, eliminar y validar ofertas. Implementa las reglas de negocio y lógica de validación como la disponibilidad de vehículos o la consistencia de fechas.

Repository (Repositorios) : Los repositorios interactúan directamente con la base de datos utilizando JPA (Java Persistence API). Definen métodos para poder realizar operaciones como CRUD (create, update, delete, etc) sobre las entidades del proecto.

Por Ejemplo:

OfertaRepository: Interfaz que extiende de JpaRepository y permite realizar operaciones con las ofertas en la base de datos.

(Entidades, Modelos): Son clases que representan los objetos de la base de datos. Cada entidad es un modelo de un recurso en el sistema, y Spring Data JPA las utiliza para mapear las tablas de la base de datos.

Ejemplo:

Oferta: Representa una oferta en la base de datos, con atributos como fechaInicio, fechaFinal, vehiculo, etc.

### **Frontend (JavaFX)**

1. **FXML (Archivos de Vista)**:
   * Los archivos .fxml son las plantillas de la interfaz gráfica. Definen la estructura visual de las ventanas y componentes, como botones, tablas, formularios, etc. Estos archivos están vinculados a los controladores de JavaFX.
   * **Ejemplo**:

OfertaView.fxml: Archivo que define la interfaz para gestionar las ofertas en la aplicación, con campos de entrada y botones para la edición o eliminación de ofertas.

1. **Controladores de FXML (JavaFX Controllers)**:
   * Los controladores son clases que manejan la lógica asociada a las vistas definidas en los archivos .fxml. Escuchan los eventos de la interfaz (como clics de botones) y ejecutan la lógica correspondiente, como llamar al backend o actualizar la interfaz.
   * **Ejemplo**:

OfertaFXMLController: Controlador encargado de gestionar la lógica detrás de la vista de ofertas. Este se comunica con el backend para realizar operaciones como editar o validar una oferta.

1. **Alert (Clase de Utilidad)**:
   * Esta clase es un singleton que se encarga de gestionar las alertas en la aplicación. Se utiliza para mostrar diferentes tipos de mensajes, como alertas de información, advertencias o errores. Centraliza la gestión de alertas, permitiendo reusar el código para mostrar mensajes al usuario.
   * **Ejemplo de método**:

mostrarAlertInfo(int code): Muestra una alerta de información dependiendo del código proporcionado. Los códigos pueden corresponder a acciones como registro exitoso, edición fallida, o campos pendientes.

### **3. Descripción de las Funcionalidades**

1. **Manejo de Ofertas**:

El sistema permite a los usuarios crear, editar y eliminar ofertas relacionadas con vehículos disponibles. Cada oferta tiene atributos como fechaInicio, fechaFinal, vehiculo, y ruta.

Crear oferta:

La interfaz permite crear una nueva oferta ingresando los detalles requeridos. El controlador JavaFX recoge los datos y los envía al backend, donde el OfertaService valida y guarda la oferta.

Editar oferta:

Los usuarios pueden modificar una oferta existente. El controlador JavaFX envía una solicitud PUT al backend para actualizar los datos. Se manejan validaciones como la disponibilidad del vehículo y la consistencia de las fechas.

Eliminar oferta:

A través de un diálogo de confirmación, el usuario puede eliminar una oferta existente. El backend elimina el registro de la base de datos y el JavaFX actualiza la interfaz.

2. Validaciones:

Disponibilidad de Vehículo:

Antes de crear o editar una oferta, el sistema verifica que el vehículo asociado esté disponible. Esta validación se realiza en el backend.

Consistencia de Fechas:

Se valida que las fechas de inicio y final de la oferta sean consistentes. Esto también es manejado por el OfertaService en el backend.

3. Alertas:

* + El sistema utiliza la clase Alert para mostrar alertas al usuario en varias situaciones, como cuando:
  + Una oferta es editada con éxito.
  + Un vehículo no está disponible.
  + Hay fechas inconsistentes en la oferta.
  + Ocurre un error al comunicarse con el backend.

**4**. Flujo de Trabajo

* Interacción del Usuario:

El usuario interactúa con la interfaz JavaFX, por ejemplo, editando una oferta a través de un formulario en la vista OfertaView.fxml.

1. Envío de Datos al Backend:

Los datos del formulario son capturados por el controlador OfertaFXMLController, el cual los envía al backend usando una petición HTTP (a través de RestTemplate)

1. Validación y Procesamiento en el Backend:

El controlador REST OfertaController en el backend recibe la petición y delega el procesamiento al OfertaService, donde se realizan validaciones (disponibilidad del vehículo, fechas, etc.).

1. Respuesta y Alertas:

El backend responde con un mensaje que indica el resultado (éxito o error). En el controlador JavaFX (OfertaFXMLController), se maneja la respuesta y se muestra la alerta correspondiente usando lAlert.

Spring Boot:

* + Para crear el backend que maneja las operaciones de negocio y provee una API REST para comunicar con el frontend.
  + JavaFX:

Para la creación de la interfaz gráfica con la que el usuario interactúa.

* + JPA / Hibernate:

Para manejar las operaciones con la base de datos de manera transparente.

* + RestTemplate:

Para que el frontend (JavaFX) se comunique con el backend a través de HTTP.