UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD POLITÉCNICA

DIRECCIÓN ACADÉMICA
DEPARTAMENTO DE POSTGRADO

Ciudad Universitaria
San Lorenzo - Paraguay

Módulo *Sistema de Información Web*

Reporte técnico **Control-financiero**

Profesor: **Julio César Mello Roman**

Alumnos: **Guillermo Palacios**
Aldo Javier Reyes

Año 2025

INDICE

Tabla de contenido

Introducción

Este reporte describe las modificaciones realizadas al proyecto “control-financiero” para incorporar dos nuevos catálogos: Países y Ciudades. Las implementaciones se realizaron siguiendo el patrón de diseño Modelo-Vista-Controlador (MVC), utilizando Spring Boot como framework principal, Hibernate para la gestión de entidades relacionales y PostgreSQL como base de datos. El entorno de desarrollo fue Eclipse, complementado con herramientas como DBeaver para la administración de la base de datos y Lombok para optimizar el código. Se crearon nuevas clases, servicios y vistas, asegurando la integración con las entidades existentes y manteniendo la escalabilidad del sistema.

2.Marco teórico

2.1Tecnologías Utilizadas

- **Framework**: Spring Boot (gestión de dependencias y configuración simplificada).
- **ORM:** Hibernate (mapeo objeto-relacional).
- **Base de Datos**: PostgreSQL, administrada mediante DBeaver.
- **Entorno de Desarrollo**: Eclipse IDE.
- **Librerías Auxiliares:** Lombok (reducción de código repetitivo).
- **Control de Versiones:** Git/GitHub para la gestión de cambios

2.2 Arquitectura del Proyecto

El proyecto sigue el patrón MVC, que organiza el código en tres capas:

- **Modelo:** Gestiona la lógica de datos y las entidades (Ciudades, Departamentos, Países).
- **Vista:** Define la interfaz de usuario mediante plantillas HTML (Thymeleaf).
- **Controlador**: Coordina las interacciones entre el modelo y la vista.
- Este enfoque garantiza un código modular, mantenible y escalable, ideal para aplicaciones empresariales como “control-financiero”.

3.Implementación técnica

3.1 Entidades y Relaciones

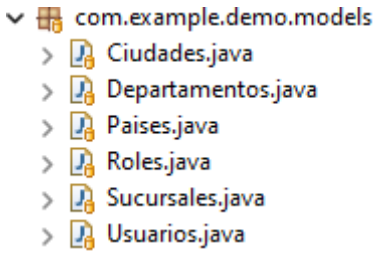
Se desarrollaron dos entidades: **Países y Ciudades**, con la siguiente relación:

- Países → Ciudades: Un país puede tener múltiples ciudades (1:N). Cada ciudad está asociada a un único país mediante una clave foránea.

3.2 Modificaciones en el Código

Las modificaciones se distribuyeron en los paquetes del proyecto, siguiendo la estructura MVC:

- Paquete model

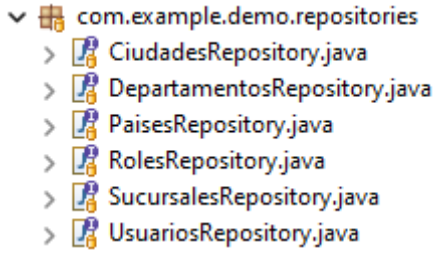


Se crearon las clases para las entidades:

- Paises.java: Define la entidad Países con atributos id_pais y descripcion.
- Ciudades.java: Define la entidad Ciudades con atributos id_ciudad, id_pais y descripcion.

Ambas clases usan anotaciones de Hibernate (@Entity, @Table, @Id, @GeneratedValue, @Column) y Lombok (@Data, @NoArgsConstructor, @AllArgsConstructor) para simplificar el código.

- Paquete repository



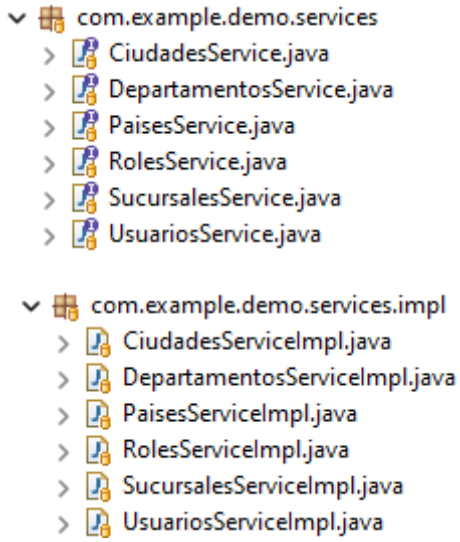
Se implementaron interfaces para operaciones CRUD:

- PaisesRepository.java: Extiende JpaRepository para Países.
- CiudadesRepository.java: Extiende JpaRepository para Ciudades.

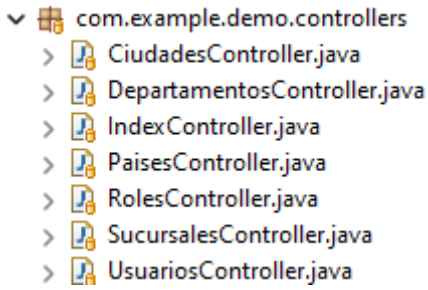
- Paquete service y service.impl

Se definieron servicios para la lógica de negocio:

- PaisesService.java / PaisesServiceImpl.java: Métodos para crear, listar, actualizar y eliminar países.
- CiudadesService.java / CiudadesServiceImpl.java: Métodos similares para ciudades, incluyendo validación de id_pais.



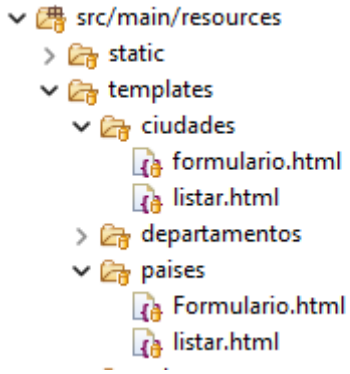
➢ Paquete controller



Se crearon controladores para manejar solicitudes HTTP:

- PaisesController.java: Gestiona endpoints como /paises/listar y /paises/formulario.
- CiudadesController.java: Gestiona endpoints como /ciudades/listar y /ciudades/formulario.

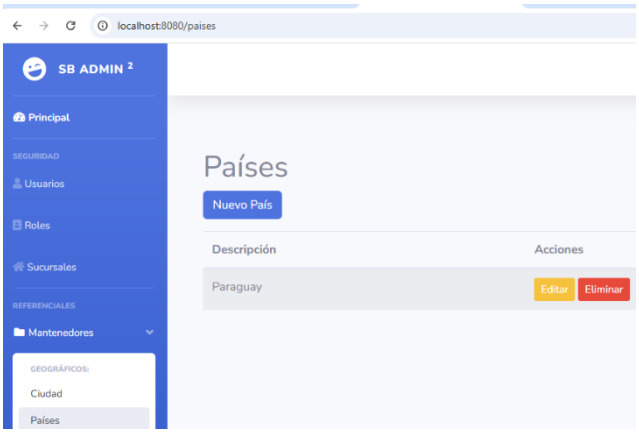
➢ Recursos (resources/templates)



Se desarrollaron vistas HTML en las carpetas:

- paises/: formulario.html (formulario para crear/editar países), listar.html (lista de países).
- ciudades/: formulario.html (formulario para crear/editar ciudades), listar.html (lista de ciudades).

3.3 Datos de Prueba



4. Flujo de Desarrollo y Versionamiento

4.1 Proceso en Git/GitHub

El desarrollo se gestionó con un flujo de trabajo basado en Git:

1. Fork: Se creó una copia del repositorio ubuntumb/control-financiero en el repositorio personal.
2. Clonación:

bash

git clone https://github.com/TU-USUARIO/control-financiero.git

cd control-financiero

3. Creación de Rama:

bash

git checkout -b name_rama

4. Desarrollo: Se implementaron los cambios en Eclipse.
5. Confirmación:

git add .

git commit -m "Implementación de catálogos Países y Ciudades"

6. Subida al Fork:

bash

git push origin feature/paises-ciudades

7. Pull Request:

Se creó un pull request desde TU-USUARIO/control-financiero:feature/paises-ciudades hacia ubuntumb/control-financiero:main.

Se incluyó un título descriptivo (e.g., “Agregado de Catálogos Países y Ciudades”) y una explicación de los cambios realizados

4.2 Comandos claves

 # Clonar

git clone https://github.com/TU-USUARIO/control-financiero.git

cd control-financiero

 # Crear rama

git checkout -b feature/paises-ciudades

 # Confirmar cambios

git add .

git commit -m "Implementación de Países y Ciudades"

 # Subir cambios

git push origin feature/paises-ciudades

5. Enlace al proyecto

https://github.com/gpcios65/control-financiero

https://github.com/ubuntumb/control-financiero/pull/5

6. Conclusión

La incorporación de los catálogos Países y Ciudades en “control-financiero” se completó exitosamente, ampliando las capacidades del sistema. El uso de Spring Boot, Hibernate y MVC garantizó un desarrollo eficiente y mantenible. Este trabajo sienta las bases para futuras mejoras, como la integración de nuevas funcionalidades geográficas.