Proyecto Final Bases de Datos
Informe de la solución realizada

David Santiago Puerto Ochoa, Jhon Sebastián Castañeda Lamprea, Johan Felipe Silva Cavieles, Juan Esteban Oyola Galindo, Sebastian Vega Baquero, Juan David Castro García.

Universidad El Bosque,Bogotá - Colombia,
dpuertoo@unbosque.edu.co,
jscastanedal@unbosque.edu.co,
joyolag@unbosque.edu.co, jfsilvac@unbosque.edu.co,
svegaba@unbosque.edu.co, jdcastrog@unbosque.edu.co.

I. Resumen

Este informe tiene como propósito informar sobre el desarrollo del proyecto destinado a establecer una base de datos funcional, la cual almacenará los datos conforme a las especificaciones del cliente y en cumplimiento con los requisitos establecidos.

La implementación se llevó a cabo utilizando el lenguaje de programación JAVA, a través de la plataforma Eclipse. Asimismo, se elaboró un diagrama de entidad-relación que representa visualmente la estructura de las tablas que componen la base de datos.

En adición, se procedió con la normalización de las tablas pertinentes, con el objetivo de prevenir posibles confusiones al momento de establecer relaciones entre las tablas propuestas.

Para concluir, se generaron los informes solicitados por el usuario, marcando así la finalización de las tareas delineadas por el equipo de trabajo.

II. Diseño de la base de datos

La base de datos se desarrolló utilizando el Framework de aplicaciones web Spring Boot en su versión 4.0. La creación de las tablas se realiza mediante la anotación @Entity, y se generan los datos necesarios, estableciendo relaciones entre las tablas mediante las anotaciones correspondientes. A continuación, se presenta un ejemplo de cómo crear una tabla utilizando Spring Boot.

```java
package co.edu.unbosque.EntreCOL.model;

import java.util.Date;

@Entity
@Table(name = "NominaEmpleado")
public class NominaEmpleado {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "idEmpleado")
    private Empleados idEmpleado;

    private boolean novedadIncapacidad;

    private boolean novedadVacaciones;

    private Integer diasTrabajados;

    private Integer diasIncapacidad;
```

Figura x. Creación de tabla usando Spring Boot.

Spring Boot se destaca por su intuitividad y facilidad de uso en la integración entre el BackEnd y el FrontEnd. A continuación, se presenta un ejemplo de cómo recibir un archivo y agregarlo a la base de datos.

```java
public class EmpleadosController {
    @Autowired
    private EmpleadosRepository daoEmpleados;
    @Autowired
    private NominaEmpleadosRepository daoNominaEmpleado;

    @PostMapping("/empleados")
    public String leerArchivo(@RequestParam("file") MultipartFile file, Model model) {
        if (file.isEmpty()) {
            return "Por favor seleccione un archivo para subir";
        }
        try {
            List<Empleados> empleados = new ArrayList<>();
            try (Workbook workbook = new XSSFWorkbook(file.getInputStream())) {
                Sheet sheet = workbook.getSheetAt(0);
                for (int i = 1; i < sheet.getPhysicalNumberOfRows(); i++) {
                    Row row = sheet.getRow(i);
                    Empleados empleado = new Empleados();
                    empleado.setCodigo((int)row.getCell(0).getNumericCellValue());
                    empleado.setNombre(row.getCell(1).getStringCellValue());
                    empleado.setDependencia(row.getCell(2).getStringCellValue());
                    empleado.setCargo(row.getCell(3).getStringCellValue());
                    empleado.setFechaIngreso((int) row.getCell(4).getNumericCellValue());
                    empleado.setEps(row.getCell(5).getStringCellValue());
```

Figura x. Agregar un archivo a la base de datos.

En otro aspecto, el FrontEnd posee la capacidad de actualizar, agregar, eliminar y leer datos, así como generar informes. A continuación, se presenta un ejemplo de su representación visual.

Figura x. Evidencia gráfica de la solución planteada

III. Descripción de las tablas

Las primeras tablas se generaron a partir de un archivo en Excel proporcionado por el cliente, que incluye dos conjuntos de datos: uno vinculado a los empleados con información como la ID, el salario, la EPS, la dependencia, entre otros; y otro relacionado con la liquidación de los empleados y sus períodos de vacaciones.

Al normalizar la segunda tabla hasta la tercera forma normal, fue esencial dividir algunos datos para crear una tabla adicional. Así, a partir del archivo de Excel, se crearon las primeras tres tablas: Vacaciones, Empleados y Liquidación.

Empleados		
PK	código	INT
	nombre	VARCHAR(50)
	dependencia	VARCHAR(50)
	cargo	VARCHAR(50)
	eps	VARCHAR(50)
	ARL	VARCHAR(50)
	pensión	VARCHAR(50)
	sueldo	INT
	fecha_Ingreso	DATE

Figura x. Tabla de empleados.

Vacaciones		
PK	id_vacaciones	INT
FK	id_empleado	INT
	inicio_vacaciones	DATE
	fin_vacaciones	DATE

Figura x. Tabla de vacaciones.

Liquidación		
FK	id_empleado	INT
	novedad_incapacidad	VARCHAR(50)
	novedad_vacaciones	VARCHAR(50)
	dias_trabajados	INT
	dias_incapacitado	INT
	dias_vacaciones	INT
	inicio_incapacidad	DATE
	fin_incapacidad	DATE
	transporte	INT
	bonificacion	INT
PK	id_liquidacion	INT

Figura x. Tabla de liquidación

Dado que el cliente proporcionó dos archivos, "movies.dat" y "books.json", fue necesario crear dos tablas adicionales para gestionar la información de libros y películas en la base de datos correspondiente. Estas tablas se crearon con base en los atributos ya presentes en los archivos, eliminando la necesidad de realizar normalización.

Libros		
PK	idLibro	INT
	autores	VARCHAR(255)
	titulo	VARCHAR(255)
	rating	VARCHAR(50)
	isbn	VARCHAR(50)
	isbn13	VARCHAR(50)
	idioma	VARCHAR(50)
	paginas	VARCHAR(50)
	totalRatings	INT
	totalReseñas	INT
	fechaPublicacion	DATE
	publicador	VARCHAR(50)

Figura x. Tabla de libros.

Peliculas		
PK	idPeliculas	INT
	nombre	VARCHAR(50)
	genero	VARCHAR(50)

Figura x. Tabla de películas.

IV. Relación entre las tablas

Las relaciones se establecieron en las primeras tres tablas (Empleados, Vacaciones, Liquidación), considerando no solo su origen común en el mismo archivo, sino también las conexiones implícitas a través de claves foráneas. La tabla de Empleados está vinculada a la tabla de Vacaciones mediante una relación de uno a uno, indicando que cada empleado puede tener solo unas vacaciones. Por otro lado, la tabla de Vacaciones se relaciona con la de Empleados a través de una relación de 'muchos', reflejando la posibilidad de aplicar las vacaciones a varios empleados simultáneamente durante un período específico.

Asimismo, tanto la tabla de Liquidación como la de Empleados están conectadas mediante una relación de uno a uno, ya que cada empleado puede tener solo una liquidación, y cada liquidación se asigna exclusivamente a un empleado.

V. Consultas importantes

Destaca algunas consultas clave que los usuarios realizarán con frecuencia. Puedes incluir ejemplos de consultas SQL.

VI. Seguridad

Breve descripción de las medidas de seguridad implementadas para proteger la integridad de los datos.

VII. Evidencia gráfica

Primera conclusión
Segunda conclusión

Referencias

[1] Primera referencia
[2] Segunda referencia