Documentación del Proyecto: Base de Datos Empresa XYZ

1. Introducción

Este documento detalla el proceso de diseño, implementación y pruebas de la base de datos para la empresa XYZ. El sistema maneja la información de usuarios, autenticación, perfiles, fidelización y actividades.

2. Requerimientos Implementados

A continuación, se presentan los módulos solicitados y su breve descripción:

- **Usuarios**: Permite registrar a los colaboradores de la empresa con datos como nombre, apellido, estado, contraseña, cargo, salario, fecha de ingreso y perfil asociado.
- **Perfiles**: Define los roles de los usuarios dentro de la empresa, con información como nombre del perfil, fecha de vigencia, descripción y encargado.
- **Login**: Registra los intentos de inicio de sesión de cada usuario, guardando la fecha y si el intento fue exitoso o fallido.
- **Fidelización**: Administra las actividades realizadas por los colaboradores, registrando su participación y acumulación de puntos.
- **Actividades**: Almacena los eventos organizados por la empresa cada 15 días en los que los empleados pueden participar y obtener puntos.
- **Vistas**: Se implementaron vistas para facilitar el análisis de la información.
- **Procedimientos Almacenados**: Se desarrollaron procedimientos para automatizar la inserción de datos y mejorar la administración de la base de datos.

3. Diseño del Modelo Entidad-Relación (EER)

Se diseñó un **Diagrama Entidad-Relación (EER)** en MySQL Workbench, representando las entidades principales y sus relaciones. A continuación, se describen las tablas y sus relaciones:

Tablas y Relaciones

- **Usuarios**: Contiene datos personales, cargo y perfil del usuario.
- **Perfiles**: Define los roles dentro de la empresa.
- **Login**: Registra intentos de autenticación de los usuarios.
- **Actividades**: Representa las actividades realizadas dentro del sistema de fidelización.
- **Fidelización**: Relaciona a los usuarios con las actividades y almacena los puntos obtenidos.

4. Creación de la Base de Datos

La base de datos fue creada en MySQL Workbench y contiene las siguientes tablas con sus respectivas relaciones y restricciones de integridad referencial.

```
CREATE DATABASE empresa_xyz;
USE empresa_xyz;
```

Creación de Tablas

Se crearon las siguientes tablas con claves primarias y foráneas:

```
CREATE TABLE perfiles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    fecha_vigencia DATE,
    descripcion TEXT,
    encargado VARCHAR(100)
);

CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    estado ENUM('Activo', 'Inactivo') DEFAULT 'Activo',
    contraseña VARCHAR(255) NOT NULL,
    cargo VARCHAR(100),
    salario DECIMAL(10,2),
    fecha_ingreso DATE,
    perfil_id INT,
    FOREIGN KEY (perfil_id) REFERENCES perfiles(id) ON DELETE SET NULL
);

CREATE TABLE login (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT,
    fecha_hora DATETIME DEFAULT CURRENT_TIMESTAMP,
    exito TINYINT(1),
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
);

CREATE TABLE actividades (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    descripcion TEXT,
    fecha DATE,
    puntos_asignados INT
);

CREATE TABLE fidelizacion (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT,
    actividad_id INT,
    puntos INT,
```

```
    fecha DATETIME,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE,
    FOREIGN KEY (actividad_id) REFERENCES actividades(id) ON DELETE CASCADE
);
```

Inserción de Datos

Inserción Masiva (Inicialmente Implementada y Eliminada)

Se implementaron procedimientos almacenados para insertar **100 registros en la tabla login** y **50 registros en la tabla fidelización** de manera masiva. Posteriormente, estos procedimientos fueron eliminados y reemplazados por inserciones individuales. A continuación, se muestra el código utilizado en la primera versión:

```
DELIMITER //
CREATE PROCEDURE insertar_registros_login()
BEGIN
    DECLARE i INT DEFAULT 1;
    DECLARE usuario_random INT;
    DECLARE fecha_random DATETIME;
    DECLARE exito_random TINYINT(1);
    WHILE i <= 100 DO
        SET usuario_random = FLOOR(1 + (RAND() * 20));
        SET fecha_random = NOW();
        SET exito_random = FLOOR(RAND() * 2);
        INSERT INTO login (usuario_id, fecha_hora, exito)
        VALUES (usuario_random, fecha_random, exito_random);
        SET i = i + 1;
    END WHILE;
END //

CREATE PROCEDURE insertar_registros_fidelizacion()
BEGIN
    DECLARE i INT DEFAULT 1;
    DECLARE usuario_random INT;
    DECLARE actividad_random INT;
    DECLARE puntos_random INT;
    DECLARE fecha_random DATETIME;
    WHILE i <= 50 DO
        SET usuario_random = FLOOR(1 + (RAND() * 20));
        SET actividad_random = FLOOR(1 + (RAND() * 15));
        SET puntos_random = FLOOR(10 + (RAND() * 91));
        SET fecha_random = NOW();
        INSERT INTO fidelizacion (usuario_id, actividad_id, puntos, fecha)
        VALUES    (usuario_random,    actividad_random,    puntos_random,
fecha_random);
        SET i = i + 1;
    END WHILE;
END //
DELIMITER ;
```

Estos procedimientos fueron eliminados con:

```
DROP PROCEDURE IF EXISTS insertar_registros_login;
```

```
DROP PROCEDURE IF EXISTS insertar_registros_fidelizacion;
```

Procedimientos Almacenados Actuales

Actualmente, la base de datos utiliza procedimientos almacenados optimizados para inserciones individuales:

```
DELIMITER //
CREATE PROCEDURE agregar_usuario(
    IN p_nombre VARCHAR(50), IN p_apellido VARCHAR(50), IN p_contraseña
VARCHAR(255),
    IN p_cargo VARCHAR(100), IN p_salario DECIMAL(10,2), IN p_fecha_ingreso
DATE, IN p_perfil_id INT)
BEGIN
    INSERT INTO usuarios (nombre, apellido, estado, contraseña, cargo,
salario, fecha_ingreso, perfil_id)
    VALUES (p_nombre, p_apellido, 'Activo', SHA2(p_contraseña, 256),
p_cargo, p_salario, p_fecha_ingreso, p_perfil_id);
END //

CREATE PROCEDURE registrar_evento(
    IN p_tipo_evento VARCHAR(20), IN p_usuario_id INT, IN p_actividad_id
INT,
    IN p_exito TINYINT, IN p_puntos INT)
BEGIN
    IF p_tipo_evento = 'login' THEN
        INSERT INTO login (usuario_id, fecha_hora, exito)
        VALUES (p_usuario_id, NOW(), p_exito);
    ELSEIF p_tipo_evento = 'fidelizacion' THEN
        INSERT INTO fidelizacion (usuario_id, actividad_id, puntos, fecha)
        VALUES (p_usuario_id, p_actividad_id, p_puntos, NOW());
    END IF;
END //
DELIMITER ;
```

Con esta optimización, los registros se realizan de manera individual según sea necesario.

8. Conclusiones y Lecciones Aprendidas

- Se logró diseñar e implementar una base de datos completa con módulos de autenticación y fidelización.
- Se aplicaron conceptos avanzados de SQL como **vistas, procedimientos almacenados y relaciones foráneas**.
- Se optimizó la inserción de datos con **procedimientos parametrizados en lugar de inserciones masivas**.