

**UNIVERSIDAD DE GUAYAQUIL  
FACULTAD DE CIENCIAS MATEMATICAS Y FISICAS**

MANUAL TECNICO

**INTEGRANTES**  
Vera Lopez Alex Gabriel

Nadia Nallely Baquerizo Avila

Myrian Patricia Yaucan Curicama

Ricardo Steven Castro Agudo

Wilson Fabricio Ortiz Aguirre

Edwin Elieser Franco Cruz

**MATERIA**

Construcción de Software

**Manual Técnico para el Sistema de Reparación de Dispositivos Móviles**

# Introducción

El proyecto "Reparación de Dispositivos Móviles" es una aplicación diseñada para gestionar y administrar el proceso de reparación de dispositivos móviles, como teléfonos, tablets, y otros equipos similares. La aplicación permite registrar clientes, técnicos, equipos, repuestos, servicios, y reparaciones, proporcionando una solución integral para la gestión de reparaciones de dispositivos móviles.

Este manual técnico proporciona información detallada sobre los requisitos del sistema, la instalación, y la configuración necesaria para ejecutar la aplicación "Reparación de Dispositivos Móviles". El objetivo es facilitar la implementación y mantenimiento de la aplicación para desarrolladores y técnicos.

Este manual está dirigido a desarrolladores, técnicos, y cualquier persona responsable de la instalación, configuración, y mantenimiento de la aplicación "Reparación de Dispositivos Móviles".

## 2. Requisitos del sistema

## Hardware

* **Mínimos:**
  + CPU: Procesador de doble núcleo a 2.0 GHz
  + RAM: 4 GB
  + Almacenamiento: 500 MB de espacio disponible
* **Recomendados:**
  + CPU: Procesador de cuadruple núcleo a 3.0 GHz
  + RAM: 8 GB o más
  + Almacenamiento: 1 GB de espacio disponible

## Software

* **Sistema operativo:**
  + Windows 10 o posterior
  + macOS 10.14 o posterior (soporte limitado)
  + Linux (distribuciones basadas en Debian o Ubuntu, soporte limitado)
* **Frameworks y dependencias:**
  + .NET Framework 4.8 o posterior
  + SQL Server 2019 o posterior
* **Otros:**
  + Visual Studio 2022 para la compilación y desarrollo
  + SQL Server Management Studio (SSMS) para la gestión de la base de datos

## Base de datos

* **Tecnología usada:**
  + Microsoft SQL Server

# 3. Instalación

## Instrucciones para configurar el entorno

### 3.1 Instalación de dependencias

1. **Instalar .NET Framework 4.8:**
   * Descargar e instalar el .NET Framework 4.8 desde el sitio web de Microsoft:
2. **Instalar SQL Server 2019:**
   * Descargar e instalar SQL Server 2019 desde el sitio web de Microsoft:
3. **Instalar SQL Server Management Studio (SSMS):**
   * Descargar e instalar SSMS desde el sitio web de Microsoft:
4. **Instalar Visual Studio 2022:**
   * Descargar e instalar Visual Studio 2022 desde el sitio web de Microsoft:

### 3.2 Configuración de la base de datos

1. **Crear una nueva base de datos:**
   * Abra SQL Server Management Studio (SSMS).
   * Conéctese a la instancia de SQL Server.
   * Haga clic con el botón derecho en "Bases de datos" y seleccione "Nueva base de datos..."
   * Asigne un nombre a la base de datos (por ejemplo, "ReparacionesMoviles") y haga clic en "Aceptar".
2. **Ejecutar scripts SQL:**
   * **Tablas:**
     + Abra el archivo SQLtablas.sql proporcionado en el proyecto.
     + Ejecute el script para crear las tablas en la base de datos.
   * **Procedimientos almacenados:**
     + Abra el archivo SQLprocedimientos.sql proporcionado en el proyecto.
     + Ejecute el script para crear los procedimientos almacenados en la base de datos.
   * **Datos de prueba:**
     + Abra el archivo SQLdatosPrueba.sql proporcionado en el proyecto.
     + Ejecute el script para insertar datos de prueba en las tablas.

### 3.3 Ejecución del servidor o aplicación

1. **Abrir el proyecto en Visual Studio 2022:**
   * Descargue el proyecto desde el repositorio GitHub:
     + [ReparacionesMovil GitHub](https://github.com/Gabo-UG/ReparacionesMovil.git)
   * Descomprima el archivo ZIP descargado.
   * Abra el archivo ReparacionesMovil.sln ubicado en la carpeta Code.
2. **Configurar la conexión a la base de datos:**
   * Abra el archivo App.config o Web.config en el proyecto.
   * Actualice la cadena de conexión para que coincida con la configuración de su base de datos.
3. **Compilar y ejecutar el proyecto:**
   * Una vez abierto el proyecto en Visual Studio.
   * Compilar el proyecto para asegurarse de que no hay errores.
   * Ejecutar el proyecto presionando F5 o utilizando el comando run.

### 

### 

### 

### 

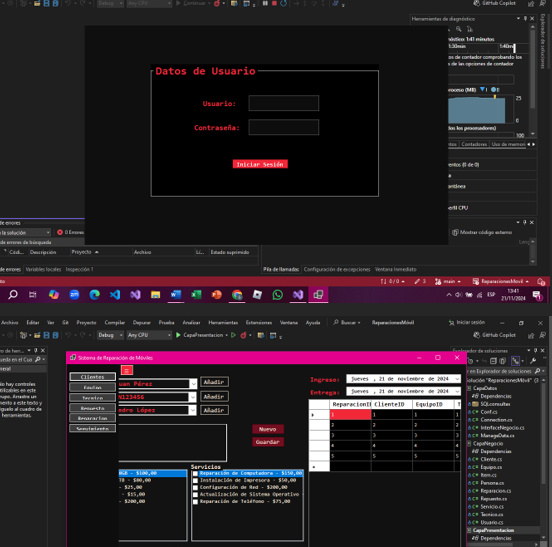
### 

### 

### 

### Archivos o scripts necesarios

* **ReparacionesMovil.sln: Archivo de solución de Visual Studio.**



Link del archivo Reparacionesmovil.sln

<https://github.com/Gabo-UG/ReparacionesMovil/blob/main/Code/ReparacionesM%C3%B3vil.sln>

* **SQLtablas.sql: Script para crear las tablas de la base de datos.**

-- Crear la tabla Cliente

CREATE TABLE Cliente (

ClienteID INT PRIMARY KEY IDENTITY,

Nombre NVARCHAR(100),

Direccion NVARCHAR(100),

Telefono NVARCHAR(20),

Correo NVARCHAR(100)

);

-- Crear la tabla Tecnico

CREATE TABLE Tecnico (

TecnicoID INT PRIMARY KEY IDENTITY,

Nombre NVARCHAR(100),

Especialidad NVARCHAR(100),

Telefono NVARCHAR(20),

Correo NVARCHAR(100)

);

-- Crear la tabla Equipo

CREATE TABLE Equipo (

EquipoID INT PRIMARY KEY IDENTITY,

Tipo NVARCHAR(100),

NumeroSerie NVARCHAR(100),

FechaIngreso DATE,

Estado NVARCHAR(50),

ClienteID INT,

FOREIGN KEY (ClienteID) REFERENCES Cliente(ClienteID)

);

-- Crear la tabla Repuesto

CREATE TABLE Repuesto (

RepuestoID INT PRIMARY KEY IDENTITY,

Nombre NVARCHAR(100),

Codigo NVARCHAR(50),

Precio DECIMAL(18, 2),

Iva DECIMAL(18, 2),

Total DECIMAL(18, 2)

);

-- Crear la tabla Servicio

CREATE TABLE Servicio (

ServicioID INT PRIMARY KEY IDENTITY,

Nombre NVARCHAR(100),

Categoria NVARCHAR(100),

Codigo NVARCHAR(50),

Precio DECIMAL(18, 2),

Iva DECIMAL(18, 2),

Total DECIMAL(18, 2)

);

-- Crear la tabla Reparacion

CREATE TABLE Reparacion (

ReparacionID INT PRIMARY KEY IDENTITY,

ClienteID INT NOT NULL,

EquipoID INT NOT NULL,

TecnicoID INT NOT NULL,

Diagnostico NVARCHAR(MAX),

FechaIngreso DATE NOT NULL,

FechaEntrega DATE NULL,

CostoTotal DECIMAL(10, 2) NULL,

FOREIGN KEY (ClienteID) REFERENCES Cliente(ClienteID),

FOREIGN KEY (EquipoID) REFERENCES Equipo(EquipoID),

FOREIGN KEY (TecnicoID) REFERENCES Tecnico(TecnicoID)

);

-- Crear la tabla ReparacionDetalle

CREATE TABLE ReparacionDetalle (

ReparacionDetalleID INT PRIMARY KEY IDENTITY(1,1),

ReparacionID INT NOT NULL,

Tipo NVARCHAR(50) NOT NULL,

ItemID INT NOT NULL,

FOREIGN KEY (ReparacionID) REFERENCES Reparacion(ReparacionID)

);

-- Crear la tabla Usuario

CREATE TABLE Usuario(

UsuarioID INT PRIMARY KEY IDENTITY(1,1),

NombreUsuario NVARCHAR(50) NOT NULL,

Contrasena NVARCHAR(50) NOT NULL

);

* **SQLprocedimientos.sql: Script para crear los procedimientos almacenados.**

-- Procedimientos para la tabla Cliente

-- Insertar Cliente

CREATE PROCEDURE InsertarCliente

@Nombre NVARCHAR(100),

@Direccion NVARCHAR(100),

@Telefono NVARCHAR(20),

@Correo NVARCHAR(100)

AS

BEGIN

INSERT INTO Cliente (Nombre, Direccion, Telefono, Correo)

VALUES (@Nombre, @Direccion, @Telefono, @Correo);

END

GO

-- Actualizar Cliente

CREATE PROCEDURE ActualizarCliente

@ClienteID INT,

@Nombre NVARCHAR(100),

@Direccion NVARCHAR(100),

@Telefono NVARCHAR(20),

@Correo NVARCHAR(100)

AS

BEGIN

UPDATE Cliente

SET Nombre = @Nombre, Direccion = @Direccion, Telefono = @Telefono, Correo = @Correo

WHERE ClienteID = @ClienteID;

END

GO

-- Eliminar Cliente

CREATE PROCEDURE EliminarCliente

@ClienteID INT

AS

BEGIN

DELETE FROM Cliente

WHERE ClienteID = @ClienteID;

END

GO

-- Seleccionar todos los Clientes

CREATE PROCEDURE SeleccionarClientes

AS

BEGIN

SELECT \* FROM Cliente;

END

GO

-- Seleccionar Cliente por ID

CREATE PROCEDURE SeleccionarClientePorID

@ClienteID INT

AS

BEGIN

SELECT \* FROM Cliente

WHERE ClienteID = @ClienteID;

END

GO

-- Procedimientos para la tabla Tecnico

-- Insertar Tecnico

CREATE PROCEDURE InsertarTecnico

@Nombre NVARCHAR(100),

@Especialidad NVARCHAR(100),

@Telefono NVARCHAR(20),

@Correo NVARCHAR(100)

AS

BEGIN

INSERT INTO Tecnico (Nombre, Especialidad, Telefono, Correo)

VALUES (@Nombre, @Especialidad, @Telefono, @Correo);

END

GO

-- Actualizar Tecnico

CREATE PROCEDURE ActualizarTecnico

@TecnicoID INT,

@Nombre NVARCHAR(100),

@Especialidad NVARCHAR(100),

@Telefono NVARCHAR(20),

@Correo NVARCHAR(100)

AS

BEGIN

UPDATE Tecnico

SET Nombre = @Nombre, Especialidad = @Especialidad, Telefono = @Telefono, Correo = @Correo

WHERE TecnicoID = @TecnicoID;

END

GO

-- Eliminar Tecnico

CREATE PROCEDURE EliminarTecnico

@TecnicoID INT

AS

BEGIN

DELETE FROM Tecnico

WHERE TecnicoID = @TecnicoID;

END

GO

-- Seleccionar todos los Tecnicos

CREATE PROCEDURE SeleccionarTecnicos

AS

BEGIN

SELECT \* FROM Tecnico;

END

GO

-- Seleccionar Tecnico por ID

CREATE PROCEDURE SeleccionarTecnicoPorID

@TecnicoID INT

AS

BEGIN

SELECT \* FROM Tecnico

WHERE TecnicoID = @TecnicoID;

END

GO

-- Procedimientos para la tabla Equipo

-- Insertar Equipo

CREATE PROCEDURE InsertarEquipo

@Tipo NVARCHAR(100),

@NumeroSerie NVARCHAR(100),

@FechaIngreso DATE,

@Estado NVARCHAR(50),

@ClienteID INT

AS

BEGIN

INSERT INTO Equipo (Tipo, NumeroSerie, FechaIngreso, Estado, ClienteID)

VALUES (@Tipo, @NumeroSerie, @FechaIngreso, @Estado, @ClienteID);

END

GO

-- Actualizar Equipo

CREATE PROCEDURE ActualizarEquipo

@EquipoID INT,

@Tipo NVARCHAR(100),

@NumeroSerie NVARCHAR(100),

@FechaIngreso DATE,

@Estado NVARCHAR(50),

@ClienteID INT

AS

BEGIN

UPDATE Equipo

SET Tipo = @Tipo, NumeroSerie = @NumeroSerie, FechaIngreso = @FechaIngreso, Estado = @Estado, ClienteID = @ClienteID

WHERE EquipoID = @EquipoID;

END

GO

-- Eliminar Equipo

CREATE PROCEDURE EliminarEquipo

@EquipoID INT

AS

BEGIN

DELETE FROM Equipo

WHERE EquipoID = @EquipoID;

END

GO

-- Seleccionar todos los Equipos

CREATE PROCEDURE SeleccionarEquipos

AS

BEGIN

SELECT \* FROM Equipo;

END

GO

-- Seleccionar Equipo por ID

CREATE PROCEDURE SeleccionarEquipoPorID

@EquipoID INT

AS

BEGIN

SELECT \* FROM Equipo

WHERE EquipoID = @EquipoID;

END

GO

-- Procedimientos para la tabla Repuesto

-- Insertar Repuesto

CREATE PROCEDURE InsertarRepuesto

@Nombre NVARCHAR(100),

@Codigo NVARCHAR(50),

@Precio DECIMAL(18, 2),

@Iva DECIMAL(18, 2),

@Total DECIMAL(18, 2)

AS

BEGIN

INSERT INTO Repuesto (Nombre, Codigo, Precio, Iva, Total)

VALUES (@Nombre, @Codigo, @Precio, @Iva, @Total);

END

GO

-- Actualizar Repuesto

CREATE PROCEDURE ActualizarRepuesto

@RepuestoID INT,

@Nombre NVARCHAR(100),

@Codigo NVARCHAR(50),

@Precio DECIMAL(18, 2),

@Iva DECIMAL(18, 2),

@Total DECIMAL(18, 2)

AS

BEGIN

UPDATE Repuesto

SET Nombre = @Nombre, Codigo = @Codigo, Precio = @Precio, Iva = @Iva, Total = @Total

WHERE RepuestoID = @RepuestoID;

END

GO

-- Eliminar Repuesto

CREATE PROCEDURE EliminarRepuesto

@RepuestoID INT

AS

BEGIN

DELETE FROM Repuesto

WHERE RepuestoID = @RepuestoID;

END

GO

-- Seleccionar todos los Repuestos

CREATE PROCEDURE SeleccionarRepuestos

AS

BEGIN

SELECT \* FROM Repuesto;

END

GO

-- Seleccionar Repuesto por ID

CREATE PROCEDURE SeleccionarRepuestoPorID

@RepuestoID INT

AS

BEGIN

SELECT \* FROM Repuesto

WHERE RepuestoID = @RepuestoID;

END

GO

-- Procedimientos para la tabla Servicio

-- Insertar Servicio

CREATE PROCEDURE InsertarServicio

@Nombre NVARCHAR(100),

@Categoria NVARCHAR(100),

@Codigo NVARCHAR(50),

@Precio DECIMAL(18, 2),

@Iva DECIMAL(18, 2),

@Total DECIMAL(18, 2)

AS

BEGIN

INSERT INTO Servicio (Nombre, Categoria, Codigo, Precio, Iva, Total)

VALUES (@Nombre, @Categoria, @Codigo, @Precio, @Iva, @Total);

END

GO

-- Actualizar Servicio

CREATE PROCEDURE ActualizarServicio

@ServicioID INT,

@Nombre NVARCHAR(100),

@Categoria NVARCHAR(100),

@Codigo NVARCHAR(50),

@Precio DECIMAL(18, 2),

@Iva DECIMAL(18, 2),

@Total DECIMAL(18, 2)

AS

BEGIN

UPDATE Servicio

SET Nombre = @Nombre, Categoria = @Categoria, Codigo = @Codigo, Precio = @Precio, Iva = @Iva, Total = @Total

WHERE ServicioID = @ServicioID;

END

GO

-- Eliminar Servicio

CREATE PROCEDURE EliminarServicio

@ServicioID INT

AS

BEGIN

DELETE FROM Servicio

WHERE ServicioID = @ServicioID;

END

GO

-- Seleccionar todos los Servicios

CREATE PROCEDURE SeleccionarServicios

AS

BEGIN

SELECT \* FROM Servicio;

END

GO

-- Seleccionar Servicio por ID

CREATE PROCEDURE SeleccionarServicioPorID

@ServicioID INT

AS

BEGIN

SELECT \* FROM Servicio

WHERE ServicioID = @ServicioID;

END

GO

-- Procedimientos para la tabla Reparacion

-- Insertar Reparacion

CREATE PROCEDURE InsertarReparacion

@ClienteID INT,

@EquipoID INT,

@TecnicoID INT,

@Diagnostico NVARCHAR(MAX),

@FechaIngreso DATE,

@FechaEntrega DATE = NULL,

@CostoTotal DECIMAL(10, 2) = NULL

AS

BEGIN

INSERT INTO Reparacion (ClienteID, EquipoID, TecnicoID, Diagnostico, FechaIngreso, FechaEntrega, CostoTotal)

VALUES (@ClienteID, @EquipoID, @TecnicoID, @Diagnostico, @FechaIngreso, @FechaEntrega, @CostoTotal);

END

GO

-- Actualizar Reparacion

CREATE PROCEDURE ActualizarReparacion

@ReparacionID INT,

@ClienteID INT,

@EquipoID INT,

@TecnicoID INT,

@Diagnostico NVARCHAR(MAX),

@FechaIngreso DATE,

@FechaEntrega DATE = NULL,

@CostoTotal DECIMAL(10, 2) = NULL

AS

BEGIN

UPDATE Reparacion

SET ClienteID = @ClienteID, EquipoID = @EquipoID, TecnicoID = @TecnicoID, Diagnostico = @Diagnostico,

FechaIngreso = @FechaIngreso, FechaEntrega = @FechaEntrega, CostoTotal = @CostoTotal

WHERE ReparacionID = @ReparacionID;

END

GO

-- Eliminar Reparacion

CREATE PROCEDURE EliminarReparacion

@ReparacionID INT

AS

BEGIN

DELETE FROM Reparacion

WHERE ReparacionID = @ReparacionID;

END

GO

-- Seleccionar todas las Reparaciones

CREATE PROCEDURE SeleccionarReparaciones

AS

BEGIN

SELECT \* FROM Reparacion;

END

GO

-- Seleccionar Reparacion por ID

CREATE PROCEDURE SeleccionarReparacionPorID

@ReparacionID INT

AS

BEGIN

SELECT \* FROM Reparacion

WHERE ReparacionID = @ReparacionID;

END

GO

-- Procedimientos para la tabla ReparacionDetalle

-- Insertar ReparacionDetalle

CREATE PROCEDURE InsertarReparacionDetalle

@ReparacionID INT,

@Tipo NVARCHAR(50),

@ItemID INT

AS

BEGIN

INSERT INTO ReparacionDetalle (ReparacionID, Tipo, ItemID)

VALUES (@ReparacionID, @Tipo, @ItemID);

END

GO

-- Actualizar ReparacionDetalle

CREATE PROCEDURE ActualizarReparacionDetalle

@ReparacionDetalleID INT,

@ReparacionID INT,

@Tipo NVARCHAR(50),

@ItemID INT

AS

BEGIN

UPDATE ReparacionDetalle

SET ReparacionID = @ReparacionID, Tipo = @Tipo, ItemID = @ItemID

WHERE ReparacionDetalleID = @ReparacionDetalleID;

END

GO

-- Eliminar ReparacionDetalle

CREATE PROCEDURE EliminarReparacionDetalle

@ReparacionDetalleID INT

AS

BEGIN

DELETE FROM ReparacionDetalle

WHERE ReparacionDetalleID = @ReparacionDetalleID;

END

GO

-- Seleccionar todos los ReparacionDetalles

CREATE PROCEDURE SeleccionarReparacionDetalles

AS

BEGIN

SELECT \* FROM ReparacionDetalle;

END

GO

-- Seleccionar ReparacionDetalle por ID

CREATE PROCEDURE SeleccionarReparacionDetallePorID

@ReparacionDetalleID INT

AS

BEGIN

SELECT \* FROM ReparacionDetalle

WHERE ReparacionDetalleID = @ReparacionDetalleID;

END

GO

-- Procedimientos para la tabla Usuario

-- Insertar Usuario

CREATE PROCEDURE InsertarUsuario

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(50)

AS

BEGIN

INSERT INTO Usuario (NombreUsuario, Contrasena)

VALUES (@NombreUsuario, @Contrasena);

END

GO

-- Actualizar Usuario

CREATE PROCEDURE ActualizarUsuario

@UsuarioID INT,

@NombreUsuario NVARCHAR(50),

@Contrasena NVARCHAR(50)

AS

BEGIN

UPDATE Usuario

SET NombreUsuario = @NombreUsuario, Contrasena = @Contrasena

WHERE UsuarioID = @UsuarioID;

END

GO

-- Eliminar Usuario

CREATE PROCEDURE EliminarUsuario

@UsuarioID INT

AS

BEGIN

DELETE FROM Usuario

WHERE UsuarioID = @UsuarioID;

END

GO

-- Seleccionar todos los Usuarios

CREATE PROCEDURE SeleccionarUsuarios

AS

BEGIN

SELECT \* FROM Usuario;

END

GO

-- Seleccionar Usuario por ID

CREATE PROCEDURE SeleccionarUsuarioPorID

@UsuarioID INT

AS

BEGIN

SELECT \* FROM Usuario

WHERE UsuarioID = @UsuarioID;

END

GO

--- faltantes

-- Procedimientos para la tabla Cliente

-- Procedimiento almacenado para obtener todos los clientes

CREATE PROCEDURE ObtenerClientes

AS

BEGIN

SELECT \* FROM Cliente;

END

GO

-- Procedimiento almacenado para obtener todos los técnicos

CREATE PROCEDURE ObtenerTecnicos

AS

BEGIN

SELECT \* FROM Tecnico;

END

GO

-- Procedimiento almacenado para obtener todos los repuestos

CREATE PROCEDURE ObtenerRepuestos

AS

BEGIN

SELECT \* FROM Repuesto;

END

GO

-- Procedimiento almacenado para obtener todos los servicios

CREATE PROCEDURE ObtenerServicios

AS

BEGIN

SELECT \* FROM Servicio;

END

GO

-- Procedimiento almacenado para consultar todas las reparaciones

CREATE PROCEDURE ConsultarTodasLasReparaciones

AS

BEGIN

SELECT \* FROM Reparacion;

END

GO

-- Procedimiento almacenado para obtener todos los equipos

CREATE PROCEDURE ObtenerEquipos

AS

BEGIN

SELECT \* FROM Equipo;

END

GO

-- Seleccionar Cliente por ID (corregido)

CREATE PROCEDURE ObtenerClientePorID

@ClienteID INT

AS

BEGIN

SELECT \* FROM Cliente

WHERE ClienteID = @ClienteID;

END

GO

-- Procedimientos para la tabla Tecnico

-- Seleccionar Tecnico por ID (corregido)

CREATE PROCEDURE ObtenerTecnicoPorID

@TecnicoID INT

AS

BEGIN

SELECT \* FROM Tecnico

WHERE TecnicoID = @TecnicoID;

END

GO

-- Procedimientos para la tabla Equipo

-- Seleccionar Equipo por Número de Serie

CREATE PROCEDURE ObtenerEquipoPorNumeroSerie

@NumeroSerie NVARCHAR(100)

AS

BEGIN

SELECT \* FROM Equipo

WHERE NumeroSerie = @NumeroSerie;

END

GO

-- Seleccionar Equipos por Cliente

CREATE PROCEDURE ObtenerEquiposPorCliente

@ClienteID INT

AS

BEGIN

SELECT \* FROM Equipo

WHERE ClienteID = @ClienteID;

END

GO

-- Seleccionar Equipos por Correo (suponiendo que se refiere al correo del cliente asociado)

CREATE PROCEDURE ObtenerEquipoPorCorreo

@Correo NVARCHAR(100)

AS

BEGIN

SELECT e.\*

FROM Equipo e

INNER JOIN Cliente c ON e.ClienteID = c.ClienteID

WHERE c.Correo = @Correo;

END

GO

-- Procedimientos para la tabla Servicio

-- Seleccionar Servicio por ID (corregido)

CREATE PROCEDURE ObtenerServicioPorID

@ServicioID INT

AS

BEGIN

SELECT \* FROM Servicio

WHERE ServicioID = @ServicioID;

END

GO

-- Procedimientos para la tabla Repuesto

-- Seleccionar Repuesto por ID (corregido)

CREATE PROCEDURE ObtenerRepuestoPorID

@RepuestoID INT

AS

BEGIN

SELECT \* FROM Repuesto

WHERE RepuestoID = @RepuestoID;

END

GO

-- Procedimientos para la tabla Reparacion

-- Crear el tipo de tabla para los detalles de reparación

CREATE TYPE DetalleReparacion AS TABLE (

Tipo NVARCHAR(50),

ItemID INT

);

GO

-- Procedimiento almacenado para insertar reparación con detalles

CREATE PROCEDURE InsertarReparacionConDetalles

@ClienteID INT,

@EquipoID INT,

@TecnicoID INT,

@Diagnostico NVARCHAR(MAX),

@FechaIngreso DATE,

@FechaEntrega DATE = NULL,

@CostoTotal DECIMAL(10, 2) = NULL,

@Detalles DetalleReparacion READONLY

AS

BEGIN

BEGIN TRANSACTION;

BEGIN TRY

-- Insertar la reparación

DECLARE @ReparacionID INT;

INSERT INTO Reparacion (ClienteID, EquipoID, TecnicoID, Diagnostico, FechaIngreso, FechaEntrega, CostoTotal)

VALUES (@ClienteID, @EquipoID, @TecnicoID, @Diagnostico, @FechaIngreso, @FechaEntrega, @CostoTotal);

SET @ReparacionID = SCOPE\_IDENTITY();

-- Insertar los detalles de la reparación

INSERT INTO ReparacionDetalle (ReparacionID, Tipo, ItemID)

SELECT @ReparacionID, Tipo, ItemID

FROM @Detalles;

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION;

THROW;

END CATCH;

END

GO

* **SQLdatosPrueba.sql: Script para insertar datos de prueba en la base de datos.**

-- Insertar datos en Cliente

INSERT INTO Cliente (Nombre, Direccion, Telefono, Correo) VALUES

('Juan Pérez', 'Calle Falsa 123', '555-1234', 'juan.perez@example.com'),

('Ana Gómez', 'Avenida Siempre Viva 742', '555-5678', 'ana.gomez@example.com'),

('Carlos Rodríguez', 'Boulevard del Libertador 456', '555-8765', 'carlos.rodriguez@example.com'),

('Laura Martínez', 'Calle Real 789', '555-4321', 'laura.martinez@example.com'),

('Luis Fernández', 'Plaza Mayor 1', '555-6789', 'luis.fernandez@example.com');

-- Insertar datos en Tecnico

INSERT INTO Tecnico (Nombre, Especialidad, Telefono, Correo) VALUES

('Pedro López', 'Electrónica', '555-2345', 'pedro.lopez@example.com'),

('Marta Ruiz', 'Mecánica', '555-3456', 'marta.ruiz@example.com'),

('José Morales', 'Programación', '555-4567', 'jose.morales@example.com'),

('Sofia Díaz', 'Redes', '555-5678', 'sofia.diaz@example.com'),

('Mario Gómez', 'Hardware', '555-6789', 'mario.gomez@example.com');

-- Insertar datos en Equipo

INSERT INTO Equipo (Tipo, NumeroSerie, FechaIngreso, Estado, ClienteID) VALUES

('Computadora', 'SN123456', '2024-01-10', 'En reparación', 1),

('Impresora', 'SN789012', '2024-02-15', 'Operativo', 2),

('Teléfono', 'SN345678', '2024-03-20', 'En reparación', 3),

('Tablet', 'SN901234', '2024-04-25', 'Operativo', 4),

('Cámara', 'SN567890', '2024-05-30', 'En reparación', 5);

-- Insertar datos en Repuesto

INSERT INTO Repuesto (Nombre, Codigo, Precio, Iva, Total) VALUES

('Memoria RAM 8GB', 'RAM8GB', 100.00, 21.00, 121.00),

('Disco Duro 1TB', 'HDD1TB', 80.00, 16.80, 96.80),

('Teclado USB', 'KBUSB', 25.00, 5.25, 30.25),

('Ratón Óptico', 'MOUSE', 15.00, 3.15, 18.15),

('Monitor 24"', 'MON24', 200.00, 42.00, 242.00);

-- Insertar datos en Servicio

INSERT INTO Servicio (Nombre, Categoria, Codigo, Precio, Iva, Total) VALUES

('Reparación de Computadora', 'Mantenimiento', 'REPCOMP', 150.00, 31.50, 181.50),

('Instalación de Impresora', 'Instalación', 'INSTIMP', 50.00, 10.50, 60.50),

('Configuración de Red', 'Consultoría', 'CONFRED', 200.00, 42.00, 242.00),

('Actualización de Sistema Operativo', 'Actualización', 'ACTOS', 120.00, 25.20, 145.20),

('Reparación de Teléfono', 'Mantenimiento', 'REPTEL', 75.00, 15.75, 90.75);

-- Insertar datos en Reparacion

INSERT INTO Reparacion (ClienteID, EquipoID, TecnicoID, Diagnostico, FechaIngreso, FechaEntrega, CostoTotal) VALUES

(1, 1, 1, 'Fallo en la placa madre', '2024-01-10', NULL, NULL),

(2, 2, 2, 'Problema con el cabezal', '2024-02-15', NULL, NULL),

(3, 3, 3, 'Software corrupto', '2024-03-20', NULL, NULL),

(4, 4, 4, 'Batería defectuosa', '2024-04-25', NULL, NULL),

(5, 5, 5, 'Error en el objetivo', '2024-05-30', NULL, NULL);

-- Insertar datos en ReparacionDetalle

INSERT INTO ReparacionDetalle (ReparacionID, Tipo, ItemID) VALUES

(1, 'Repuesto', 1),

(2, 'Servicio', 1),

(3, 'Repuesto', 2),

(4, 'Servicio', 3),

(5, 'Repuesto', 3);

-- Insertar datos en Usuario

INSERT INTO Usuario (NombreUsuario, Contrasena) VALUES

('admin', 'admin123'),

('usuario1', 'pass123'),

('usuario2', 'pass456'),

('usuario3', 'pass789'),

('usuario4', 'pass000');

-- Visualizar datos en Cliente

SELECT \* FROM Cliente;

-- Visualizar datos en Tecnico

SELECT \* FROM Tecnico;

-- Visualizar datos en Equipo

SELECT \* FROM Equipo;

-- Visualizar datos en Repuesto

SELECT \* FROM Repuesto;

-- Visualizar datos en Servicio

SELECT \* FROM Servicio;

-- Visualizar datos en Reparacion

SELECT \* FROM Reparacion;

-- Visualizar datos en ReparacionDetalle

SELECT \* FROM ReparacionDetalle;

-- Visualizar datos en Usuario

SELECT \* FROM Usuario;

**4. Arquitectura del Sistema**

La arquitectura del sistema está diseñada bajo el patrón **N-Capas**, un estándar ampliamente utilizado que separa la lógica en diferentes capas, cada una con una responsabilidad bien definida. Esto facilita la escalabilidad, el mantenimiento y el entendimiento del sistema por parte de los desarrolladores.

**Capa de Presentación (Vista)**

* **Rol principal**: Proporcionar interfaces gráficas para que los usuarios interactúen con el sistema.
* **Componentes clave**:
  + Formularios en WinForms (C#), como frmGestionCliente.cs, frmLoginUsuario.cs, etc.
  + Los formularios son responsables de capturar datos de los usuarios, validarlos de forma básica y enviarlos a la capa de negocio.
  + Ejemplo:
    - **Formulario frmGestionCliente.cs**: Permite registrar clientes, mostrando campos como nombre, dirección, teléfono y correo.
    - **Evento Click del botón Guardar**:

private void btnGuardar\_Click(object sender, EventArgs e)

{

Cliente nuevoCliente = new Cliente(txtNombre.Text, txtDireccion.Text, txtTelefono.Text, txtCorreo.Text);

if (nuevoCliente.Registrar())

{

MessageBox.Show("Cliente registrado exitosamente");

}

else

{

MessageBox.Show("Error al registrar el cliente");

}

}

**Capa de Negocio (Lógica de Aplicación)**

* **Rol principal**: Implementar las reglas del negocio y procesar los datos entre la capa de presentación y la capa de datos.
* **Componentes clave**:
  + **Clases**:
    - cliente.cs: Maneja las operaciones relacionadas con clientes, como registrar, actualizar o eliminar.
    - reparacion.cs: Controla las reparaciones, permitiendo asignar técnicos, calcular costos y gestionar estados.
    - usuario.cs: Gestiona el inicio de sesión y la asignación de roles.
  + **Ejemplo de la clase cliente.cs**:

public class Cliente

{

public int Id { get; set; }

public string Nombre { get; set; }

public string Direccion { get; set; }

public string Telefono { get; set; }

public string Correo { get; set; }

public bool Registrar()

{

// Llamada al procedimiento almacenado InsertarCliente

string query = "EXEC InsertarCliente @Nombre, @Direccion, @Telefono, @Correo";

// Ejecutar query con los parámetros correspondientes (implementación simplificada)

return DatabaseHelper.ExecuteNonQuery(query, new { Nombre, Direccion, Telefono, Correo });

}

}

**Capa de Datos (Modelo)**

* **Rol principal**: Gestionar la interacción con la base de datos mediante procedimientos almacenados y tablas.
* **Scripts incluidos**:
  + sqltablas.sql: Define la estructura de la base de datos, como las tablas de Cliente, Técnico, Reparación, etc.
  + sqlprocedimientos.sql: Contiene los procedimientos almacenados que encapsulan las operaciones de datos.
  + sqldatospruebas.sql: Inserta datos iniciales para realizar pruebas.
* **Ejemplo de procedimiento almacenado (InsertarCliente)**:

CREATE PROCEDURE InsertarCliente

@Nombre NVARCHAR(100),

@Direccion NVARCHAR(100),

@Telefono NVARCHAR(20),

@Correo NVARCHAR(100)

AS

BEGIN

INSERT INTO Cliente (Nombre, Direccion, Telefono, Correo)

VALUES (@Nombre, @Direccion, @Telefono, @Correo);

END;

**Comunicación entre las capas**

* **Flujo general**:
  1. El usuario interactúa con un formulario (capa de presentación).
  2. Los datos capturados son enviados a una clase en la capa de negocio para validación.
  3. La clase llama a un procedimiento almacenado en la capa de datos.
  4. Los resultados son devueltos y mostrados al usuario en la interfaz.

**5. Funcionalidades del Sistema**

El sistema implementa una serie de funcionalidades diseñadas para cubrir el ciclo completo de la reparación de dispositivos móviles. A continuación, se detalla cada funcionalidad:

**1. Gestión de Usuarios**

* **Descripción**: Permite gestionar el acceso al sistema y asignar roles (administrador, técnico, cliente).
* **Operaciones soportadas**:
  + Registro, inicio de sesión y eliminación de usuarios.
  + Roles definidos:
    - **Administrador**: Acceso total.
    - **Técnico**: Puede gestionar reparaciones y ver inventario.
    - **Cliente**: Consulta el estado de su reparación.
* **Ejemplo de implementación**:
  + **Formulario frmLoginUsuario.cs**:
    - Captura usuario y contraseña, valida contra la base de datos con el procedimiento SeleccionarUsuarioPorID.
    - Muestra un mensaje de error si las credenciales son incorrectas.

**2. Gestión de Clientes**

* **Descripción**: Permite registrar clientes y actualizar sus datos.
* **Procedimientos almacenados**:
  + InsertarCliente, ActualizarCliente, SeleccionarClientePorID.
* **Ejemplo**:
  + En el formulario frmGestionCliente.cs, se muestran los datos de clientes en un DataGridView y se permite agregar nuevos registros.

**3. Gestión de Equipos**

* **Descripción**: Administra los dispositivos ingresados para reparación, registrando su estado y datos asociados.
* **Ejemplo**:
  + Un cliente deja un equipo con un problema específico, como una pantalla rota.
  + El formulario frmGestionEquipos.cs captura el tipo de equipo, número de serie y cliente asociado.

**4. Gestión de Reparaciones**

* **Descripción**: Maneja el ciclo completo de una reparación, desde la recepción del equipo hasta su finalización.
* **Características destacadas**:
  + Asignación de técnicos basada en disponibilidad y especialización.
  + Registro de diagnósticos, fechas y costos.
* **Ejemplo**:
  + El formulario frmGestionReparacion.cs permite asignar un técnico a una reparación específica usando el procedimiento InsertarReparacion.

**5. Gestión de Repuestos**

* **Descripción**: Controla el inventario de repuestos y registra su uso en reparaciones.
* **Características**:
  + Reducción automática del stock al usar un repuesto.
  + Generación de alertas cuando el stock es bajo.

### ****6. Estructura del Código****

La organización del código es fundamental para garantizar su claridad y mantenimiento. A continuación, se detalla cada carpeta y archivo del proyecto:

/Modelo

sqltablas.sql

sqlprocedimientos.sql

sqldatospruebas.sql

/CapaNegocio

cliente.cs

equipo.cs

reparacion.cs

tecnico.cs

usuario.cs

/Vista

frmGestionCliente.cs

frmLoginUsuario.cs

frmMenu.cs

frmGestionReparacion.cs

**Capa de Datos**

* Scripts para la creación de tablas y procedimientos almacenados.
* Proveen una estructura clara para operaciones CRUD.

**Capa de Negocio**

* Cada clase representa una entidad del sistema (Cliente, Técnico, Reparación, etc.).
* Ejemplo de método en usuario.cs para iniciar sesión:

public bool IniciarSesion(string usuario, string contrasena)

{

string query = "EXEC ValidarUsuario @Usuario, @Contrasena";

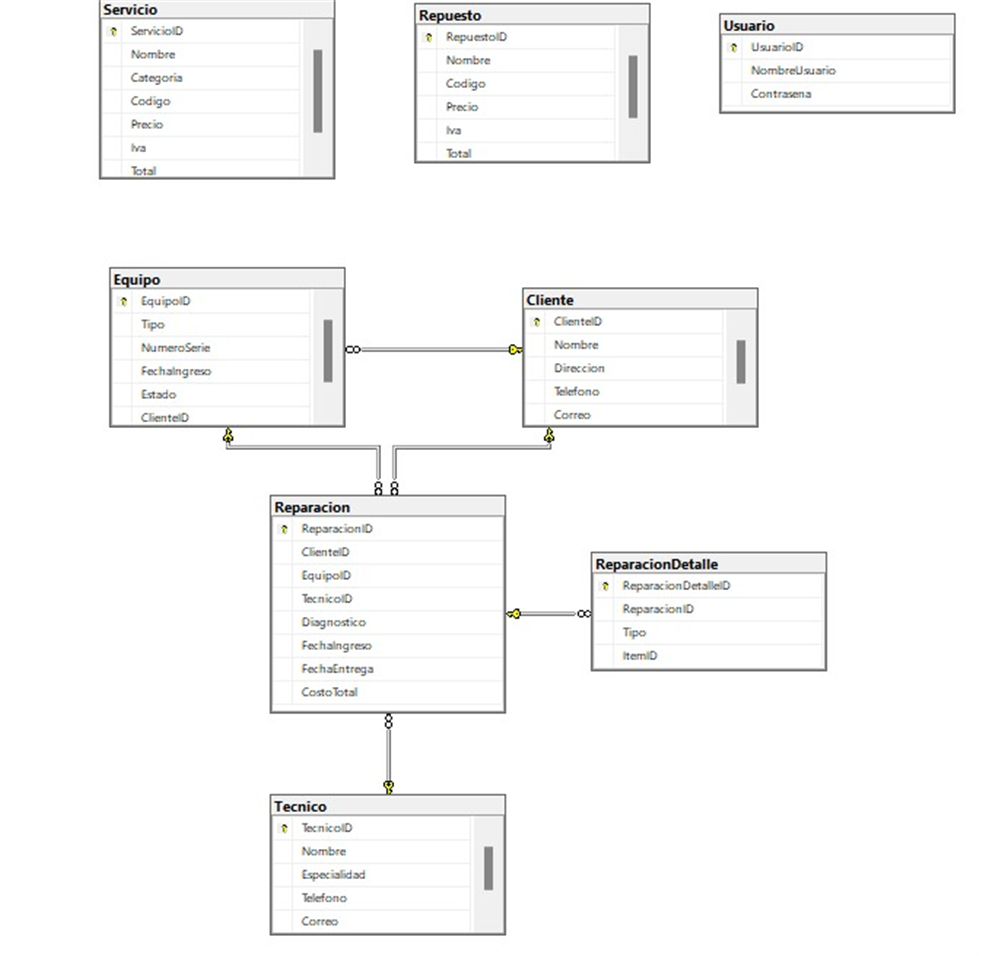
return DatabaseHelper.ExecuteScalar(query, new { Usuario = usuario, Contrasena = contrasena });

}

**Capa de Presentación**

* Formularios diseñados en WinForms para gestionar las operaciones principales.
* Uso de componentes como DataGridView, botones y cuadros de texto para interactuar con los usuarios.

## Diseño de la base de datos



# Pruebas funcionales

Las pruebas funcionales se realizarán para verificar que cada módulo del sistema de gestión de reparación de equipos móviles cumpla con los requisitos especificados y proporcione las funcionalidades necesarias para el flujo operativo. A continuación, se describe cada caso de prueba según los módulos y funcionalidades claves identificados en el alcance del proyecto.

|  |  |
| --- | --- |
| ***Código*** | *CP-01* |
| ***Caso de prueba*** | *Registro de usuario* |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Validar que el sistema permita el registro de nuevos usuarios, solicitando datos como nombre, correo electrónico y rol asignado (administrador, técnico o cliente). |
| ***Requisito previo*** | *Acceso al sistema con permisos administrativos* |
| ***Resultado esperado*** | *El usuario se registra exitosamente y es visible en el listado de usuarios* |
| ***Resultado obtenido*** | *Se ha registrado correctamente el usuario.* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *CP-02* |
| ***Caso de prueba*** | *Inicio de sesión* |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Verificar que los usuarios puedan iniciar sesión utilizando su correo electrónico y contraseña, y que el sistema aplique restricciones de acceso según el rol asignado. |
| ***Requisito previo*** | Usuarios previamente registrados en la base de datos. |
| ***Resultado esperado*** | El usuario accede al sistema y tiene permisos limitados según su rol; mensaje de error si las credenciales son incorrectas. |
| ***Resultado obtenido*** | *El usuario inicio sesión correctamente* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *CP-03* |
| ***Caso de prueba*** | Registro de Cliente |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Comprobar que se puedan registrar nuevos clientes, ingresando datos relevantes como nombre, teléfono, dirección y correo electrónico. |
| ***Requisito previo*** | Acceso autorizado al módulo de clientes. |
| ***Resultado esperado*** | El cliente se registra correctamente y es visible en la lista de clientes. |
| ***Resultado obtenido*** | *Se ha registrado correctamente al cliente.* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *CP-04* |
| ***Caso de prueba*** | Actualización de Información de Cliente |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Validar que la información de un cliente existente (como teléfono o dirección) se pueda actualizar sin problemas. |
| ***Requisito previo*** | Cliente previamente registrado en el sistema. |
| ***Resultado esperado*** | Los datos actualizados del cliente se reflejan en el sistema y en los reportes asociados. |
| ***Resultado obtenido*** | *Se ha actualizado correctamente la información del cliente.* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *CP-05* |
| ***Caso de prueba*** | Registro de Equipo en Reparación |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Asegurar que el sistema permita registrar equipos que ingresan para reparación, capturando detalles como número de serie, tipo de equipo y descripción del problema reportado. |
| ***Requisito previo*** | Acceso autorizado al módulo de equipos. |
| ***Resultado esperado*** | El equipo se registra en el sistema y queda asociado al cliente correspondiente. |
| ***Resultado obtenido*** | *Se ha registrado el equipo* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *CP-06* |
| ***Caso de prueba*** | Asignación de Técnico a Reparación |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Comprobar que se puede asignar un técnico disponible a la reparación de un equipo específico, incluyendo la revisión de su especialidad y disponibilidad. |
| ***Requisito previo*** | Técnicos registrados y disponibles en el sistema. |
| ***Resultado esperado*** | El técnico queda asignado a la reparación y esta información se almacena para seguimiento. |
| ***Resultado obtenido*** | *Se ha asignado técnico* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *CP-07* |
| ***Caso de prueba*** | Control de Inventario de Repuestos |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Verificar que el inventario de repuestos se actualice correctamente al utilizar un repuesto en una reparación, reflejando el stock actual y notificando niveles bajos. |
| ***Requisito previo*** | Registro de repuestos en el inventario del sistema. |
| ***Resultado esperado*** | El inventario se actualiza y se genera una notificación si el stock de un repuesto alcanza el nivel mínimo. |
| ***Resultado obtenido*** | *Inventario actualizado correctamente* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *CP-08* |
| ***Caso de prueba*** | generación de Factura de Servicio |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Validar que el sistema permita generar una factura detallada para cada reparación completada, incluyendo datos del cliente, técnicos involucrados, repuestos y servicios aplicados, así como el costo total. |
| ***Requisito previo*** | Equipos con reparación finalizada y repuestos aplicados. |
| ***Resultado esperado*** | La factura muestra todos los detalles de la reparación, con el desglose de costos y totales calculados correctamente. |
| ***Resultado obtenido*** | *Factura generada correctamente* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

### Pruebas de integración

Las pruebas de integración se realizan para verificar que los módulos del sistema interactúan correctamente y que las transferencias de datos entre ellos funcionan sin problemas. Estas pruebas son críticas para asegurar que los flujos de trabajo entre los módulos se ejecuten adecuadamente y que el sistema opere como una unidad cohesiva.

|  |  |
| --- | --- |
| ***Código*** | *PI-01* |
| ***Caso de prueba*** | Integración entre Módulo Usuario y Cliente |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Verificar que, al crear un usuario de tipo cliente, este se registre automáticamente en el módulo de clientes, con acceso limitado según su rol. |
| ***Requisito previo*** | Usuario de tipo cliente creado en el módulo de usuarios. |
| ***Resultado esperado*** | El cliente se registra en el módulo de clientes y tiene permisos adecuados según su rol. |
| ***Resultado obtenido*** | *El usuario tiene acceso según sus roles establecidos* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *PI-02* |
| ***Caso de prueba*** | Integración entre Cliente y Registro de Equipos |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Comprobar que los equipos registrados en el módulo de equipos se asocien correctamente al cliente correspondiente en el módulo de clientes. |
| ***Requisito previo*** | Cliente existente en el sistema y equipo registrado. |
| ***Resultado esperado*** | El equipo se vincula al cliente, y es visible en su perfil de cliente |
| ***Resultado obtenido*** | *Equipos asociados correctamente al cliente* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *PI-03* |
| ***Caso de prueba*** | Integración entre Técnico y Asignación de Reparaciones |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Validar que, al asignar un técnico a una reparación en el módulo de equipos, el técnico quede registrado en el historial de la reparación en el perfil del equipo y del cliente. |
| ***Requisito previo*** | Técnico registrado y reparación creada para un equipo. |
| ***Resultado esperado*** | El técnico se asigna correctamente y aparece en el historial de reparación del equipo y del cliente. |
| ***Resultado obtenido*** | *Detalles de la reparación y técnico asignados correctamente* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *PI-04* |
| ***Caso de prueba*** | Integración entre Repuestos y Generación de Factura |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Verificar que, al utilizar repuestos en una reparación, estos se incluyan automáticamente en la factura del cliente, junto con los costos de mano de obra del técnico. |
| ***Requisito previo*** | Repuestos asignados a la reparación y generación de factura. |
| ***Resultado esperado*** | La factura incluye todos los repuestos utilizados y calcula correctamente los costos totales. |
| ***Resultado obtenido*** | *Detalles de factura generados correctamente* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

|  |  |
| --- | --- |
| ***Código*** | *PI-05* |
| ***Caso de prueba*** | Integración entre Módulo de Seguridad y Acceso a Funcionalidades |
| ***Responsable*** | *Desarrolladores* |
| ***Descripción de la prueba*** | Asegurar que los permisos asignados a roles en el módulo de usuarios controlen el acceso a las funcionalidades del sistema, restringiendo o permitiendo operaciones según el rol del usuario. |
| ***Requisito previo*** | Roles y permisos configurados en el sistema. |
| ***Resultado esperado*** | Los usuarios solo pueden acceder a funcionalidades permitidas por su rol, evitando accesos no autorizados. |
| ***Resultado obtenido*** | *Roles asignados correctamente* |
| ***Estado*** | *Exitoso* |
| ***Observaciones*** |  |

## Herramientas de apoyo para la ejecución de pruebas

La siguiente tabla define las herramientas utilizadas para soportar el proceso de pruebas de este plan.

|  |  |  |
| --- | --- | --- |
| **Elemento de software** | **Versión** | **Tipo** |
| Selenium | Ultima Versión | Herramienta de automatización de pruebas funcionales y de interfaz de usuario. |
| JMeter | Ultima Versión | Herramienta de automatización de pruebas de rendimiento y de carga. |
| Postman | Ultima Versión | Herramienta para pruebas de API. |
| Excel | 2019 | Administración de pruebas y documentación de resultados. |
| GanttProject | 3.3 | Administración del Proyecto y seguimientos de caso de prueba. |

### Caso de prueba Protocolo de pruebas

Este documento contiene el conjunto de casos de prueba desarrollados y ejecutados, con el objetivo de validar cada módulo del sistema de gestión de reparación de equipos móviles. Incluye la siguiente información:

* **Identificación de los Casos de Prueba:** Código único asignado a cada caso de prueba, con una descripción detallada de la funcionalidad evaluada en cada módulo, tales como:
  + - **Módulo de Usuario:** Casos de prueba para registro, autenticación, asignación de roles y permisos.
    - **Módulo de Clientes:** Validación de la creación, actualización, eliminación y consulta de clientes.
    - **Módulo de Equipos**: Registro, actualización y eliminación de equipos, además del cambio de estado en el proceso de reparación.
    - **Módulo de Técnicos:** Asignación de técnicos a reparaciones, actualización de datos y consulta de historial de actividad.
    - **Módulo de Repuestos:** Control de inventario, registro y consulta de repuestos utilizados en las reparaciones.
    - **Módulo de Facturación:** Generación de facturas detalladas con los servicios y repuestos utilizados.
* **Pasos para la Ejecución:** Instrucciones detalladas para ejecutar cada caso de prueba, incluyendo secuencia de navegación, interacciones y valores de entrada específicos.
* **Datos de Prueba:** Valores utilizados en cada caso, tales como datos de cliente, equipo, repuestos y técnico, para simular un entorno de prueba realista.
  + - Resultados Esperados y Obtenidos: Comparación entre el resultado esperado y el observado para cada caso. Ejemplo:
    - Resultado Esperado: Un cliente registrado aparece en la lista de clientes con todos sus datos correctamente guardados.
    - Resultado Obtenido: El cliente aparece en la lista sin datos omitidos, confirmando la funcionalidad del registro.
* **Estado de la Prueba:** Resultado de cada caso, especificando si fue Exitoso o Fallido.

# Mantenimiento

El mantenimiento del sistema es esencial para garantizar un rendimiento adecuado en el tiempo. Las actividades y protocolos necesarios para mantener un sistema de reparación de dispositivos móviles se describen a continuación.

## Actualizaciones del sistema

El proceso de actualización le permite incluir nuevas funciones, corregir errores o mejorar el rendimiento del sistema. Para actualizar, siga estos pasos:

Haga una copia de seguridad de sus datos: es importante hacer una copia de seguridad de su base de datos y de los archivos críticos del sistema antes de realizar cualquier actualización.

Base de datos: utilice herramientas de exportación o scripts SQL para crear una copia de seguridad completa.

Archivos del sistema: asegúrese de que todos los archivos de configuración y datos estén respaldados.

## Implementación de actualización:

Descargue u obtenga una nueva versión del software.

Siga las instrucciones de instalación proporcionadas para aplicar la actualización (que pueden incluir actualizaciones dependientes, cambios de código, etc.).

Si es necesario, utilice el script de migración para hacer coincidir la información de estado con la nueva versión.

**Controlar:**

Después de realizar la actualización, verifique que el sistema esté funcionando correctamente realizando pruebas de regresión.

Confirma la nueva funcionalidad y asegúrate de que la original sigue funcionando sin problemas.