DPTO. INFORMÁTICA - I.E.S. LA MARISMA

MÓDULO PROYECTO

C.F.G.S.
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

FichAPP

Autor/es: EMILIO RAMÍREZ JOSÉ

Fecha: 24/042025

Tutora: AGUEDA MARIA LOPEZ MORENO

FICHA DEL TRABAJO FINAL

TÍTULO DEL TRABAJO	**FichAPP**
AUTOR: EMILIO RAMÍREZ JOSÉ	**FECHA: 10/06/2025**
TUTOR: AGUEDA MARIA LOPEZ MORENO	
TITULACIÓN	**CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMAS**
PALABRAS CLAVE	**Fichaje, Control horario, Registro de jornada, Asistencia, Tiempo de trabajo, Marcaje, Entrada y salida, Puntualidad**

RESUMEN DEL PROYECTO.

(Español) Esta aplicación sirve como gestor de fichas de los trabajadores de una empresa. Se encarga de determinar cuando entra un trabajador y cuando termina, si entra más tarde re calcula el tiempo que debe, también calcula el número de pausas que hace el trabajador.

(English) This application serves as an employee record manager. It keeps track of when a worker starts and finishes their shift. If they arrive late, it recalculates the time they owe. It also tracks the number of breaks the employee takes.

1. INTRODUCCIÓN

1.1. INTRODUCCIÓN A LA MEMORIA.

Este documento describe el desarrollo del proyecto Sistema de Gestión de Asistencia Laboral con monitorización, una aplicación de escritorio diseñada para automatizar el registro de entradas, salidas y pausas de los trabajadores, garantizando precisión y transparencia mediante el uso de tecnología Windows Forms . La memoria detalla el proceso de análisis, diseño, implementación y pruebas.

1.2. DESCRIPCIÓN.

El sistema permite:

- Fichaje inteligente: Registro de horarios con validación por geolocalización (solo dentro del área laboral).
- Cálculo automático: Compensación de retrasos y control de pausas activas.
- Panel administrativo: Visualización de datos y generación de informes.Está dirigido a pequeñas y medianas empresas que buscan eliminar errores manuales y optimizar la gestión de asistencia.

1.3. OBJETIVOS GENERALES.

- Desarrollar una aplicación web funcional que registre fichajes con geolocalización.
- Implementar autenticación segura por roles (empleado vs. administrador).
- Garantizar precisión en el cálculo de horas trabajadas y pausas

1.4. BENEFICIOS.

- Para empresas:
  - Reducción de fraudes en fichajes.
  - Ahorro de tiempo en gestión manual.
  - Informes automatizados para nóminas.
- Para trabajadores:
  - Transparencia en el registro de su jornada.
  - Accesibilidad desde cualquier dispositivo.

1.5. MOTIVACIONES PERSONALES.

Este proyecto nace de:

- Interés por la tecnología aplicada a problemas reales: Solucionar un proceso tedioso (control de asistencia) con herramientas modernas.
- Deseo de aprender: Profundizar en desarrollo de aplicaciones de escritorio y en el uso de elementos de monitorización.

1.6. ESTRUCTURA DE LA MEMORIA

El documento se organiza en:

- Introducción (contexto y objetivos).
- Estudio de viabilidad (análisis técnico y económico).
- Análisis (requisitos y diagramas UML).
- Diseño (arquitectura y tecnologías).
- Implementación (código y funcionalidades clave).
- Pruebas (casos y resultados).
- Conclusiones (logros y mejoras futuras).
- Anexos (manuales, código relevante).

2. ESTUDIO DE VIABILIDAD

2.1. INTRODUCCIÓN

2.1.1. TIPOLOGÍA Y PALABRAS CLAVE

- Tipología: Aplicación de escritorio para gestión de fichajes laborales.
- Palabras clave:
  - **Control de asistencia**
  - **C#**
  - **SQL Server**
  - **Visual Studio**
  - **Aplicación Windows**

2.1.2. DESCRIPCIÓN

Sistema de escritorio desarrollado en C# con .NET Framework que permite registrar la entrada, salida y pausas de los trabajadores. Utiliza SQL Server como base de datos para almacenar los registros de manera segura y eficiente.

2.1.3. OBJETIVOS DEL PROYECTO

- Automatizar el fichaje laboral mediante una interfaz intuitiva.
- Validar la identidad del trabajador (usuario/contraseña).
- Generar informes de horas trabajadas, retrasos y pausas.
- Garantizar la persistencia de datos con SQL Server.

2.1.4. CLASIFICACIÓN DE LOS OBJETIVOS

Tipo	Objetivo
Funcional	Registro de fichajes y generación de informes
Técnico	Desarrollo en C# con SQL Server
Económico	Coste mínimo (licencias gratuitas para estudiantes)

2.1.5. DEFINICIONES, ACRÓNIMOS Y ABREVIACIONES

- SQL Server: Sistema de gestión de bases de datos relacionales de Microsoft.
- .NET Framework: Entorno de desarrollo para aplicaciones Windows.
- C#: Lenguaje de programación orientado a objetos.

2.1.6. PARTES INTERESADAS

- Empresas: Para controlar la asistencia de empleados.
- Trabajadores: Usuarios finales que registran su jornada.
- Departamento de RRHH: Gestión de nóminas y horarios.

2.1.7. REFERENCIAS

- Documentación de Microsoft SQL Server.
- Guía oficial de C# y .NET.

2.1.8. DOCUMENTACIÓN DEL PROYECTO

Incluirá:

- Manual de usuario (instrucciones de uso).
- Manual técnico (instalación y configuración de SQL Server).
- Memoria del proyecto.

2.2. ESTUDIO DE LA SITUACIÓN ACTUAL

2.2.1. CONTEXTO

Muchas empresas aún utilizan sistemas de fichaje manuales (hojas de cálculo o relojes físicos), lo que genera errores y pérdida de tiempo. Este proyecto ofrece una solución digital integrada en el entorno Windows.

2.2.2. LÓGICA DEL SISTEMA

- Login del trabajador (autenticación con SQL Server).
- Registro de entrada/salida (timestamp en la base de datos).
- Cálculo automático de horas trabajadas y pausas.
- Generación de informes para RRHH.

2.2.3. DESCRIPCIÓN FÍSICA

- Front-end: Aplicación Windows Forms o WPF en C#.
- Back-end: SQL Server para almacenamiento de datos.
- Entorno de desarrollo: Visual Studio Community (gratuito).

2.2.4. DIAGNÓSTICO DEL SISTEMA ACTUAL

- Problemas detectados:
  - Fichajes manuales inexactos.
  - Falta de integración con sistemas de nóminas.
- Oportunidades:
  - Mayor precisión y trazabilidad con SQL Server.

2.2.5. NORMATIVA Y LEGISLACIÓN

- Cumple con el Reglamento General de Protección de Datos (RGPD).
- Los datos se almacenan de forma segura en SQL Server (encriptación opcional).

2.3. REQUISITOS DEL SISTEMA

2.3.1. REQUISITOS

- Funcionales:
  - RF1: Login con autenticación en base de datos.
  - RF2: Registro de fichajes con marca de tiempo.
  - RF3: Generación de informes en PDF/Excel.
- No funcionales:
  - RNF1: Compatibilidad con Windows 10/11.
  - RNF2: Tiempo de respuesta < 1 segundo en operaciones locales.

2.3.2. RESTRICCIONES DEL SISTEMA

- Requiere SQL Server instalado en el equipo o servidor.
- Solo funciona en entornos Windows.

2.3.3. CATALOGACIÓN Y PRIORIZACIÓN DE LOS REQUISITOS

Prioridad	Requisito
Alta	RF1, RF2
Media	RD3
Baja	RNF1 (compatibilidad)

2.4. ALTERNATIVAS Y SELECCIÓN DE LA SOLUCIÓN

2.4.1. ALTERNATIVA 1: C# + SQL Server

- Ventajas:
  - Alto rendimiento y seguridad.
  - Ideal para entornos empresariales Windows.
- Inconvenientes:
  - Requiere licencia de SQL Server para producción (no para desarrollo).

2.4.2. ALTERNATIVA 2: C# + SQLite

- Ventajas:
  - Base de datos ligera y portable.
- Inconvenientes:
  - Limitada para múltiples usuarios concurrentes.

2.4.3. CONCLUSIONES

Se elige Alternativa 1 (SQL Server) por su escalabilidad y compatibilidad con entornos profesionales.

2.5. PLANIFICACIÓN DEL PROYECTO

2.5.1. RECURSOS DEL PROYECTO

- Humanos: 1 desarrollador (el alumno).
- Tecnológicos: Visual Studio Community, SQL Server Express (gratis).

2.5.2. TAREAS DEL PROYECTO

- Diseño de la base de datos (Semana 1).
- Desarrollo de la interfaz en C# (Semana 2-3).

- Conexión a SQL Server (Semana 4).
- Pruebas y documentación (Semana 5).

2.5.3. PLANIFICACIÓN TEMPORAL

Tarea	Duración
Análisis y diseño de BD	1 semana
Desarrollo Front-End	2 semanas
Integración con SQL Server	1 semanas
Pruebas y depuración	1 semana

2.6. EVALUACIÓN DE RIESGOS

2.6.1. LISTA DE RIESGOS

- Errores en la conexión a SQL Server.
- Problemas de rendimiento con grandes volúmenes de datos.

2.6.2. CATALOGACIÓN DE RIESGOS

Riesgo	Probabilidad	Impacto
Conexión a SQL Server	Bajo	Medio
Rendimiento	Baja	Bajo

2.6.3. PLAN DE CONTINGENCIA

- Para conexiones: Usar try-catch y mensajes de error descriptivos.
- Para rendimiento: Optimizar consultas SQL con índices.

2.7. PRESUPUESTO

2.7.1. ESTIMACIÓN DE COSTE MATERIAL

- Software:
  - Visual Studio Community: 0€.

- SQL Server Management Studio 20: 0€.
- Hardware: Ordenador estándar (ya disponible).

2.7.2. ESTIMACION DE COSTE PERSONAL

- Horas estimadas: 50 horas (valoradas a 8€/hora como becario) = 400€.

2.7.3. RESUMEN Y ANÁLISIS COSTE BENEFICIO

- Coste total: 400€ (solo mano de obra).
- Beneficios:
  - Elimina costes de sistemas de fichaje físico.
  - Reducción de errores en nóminas.

2.8. CONCLUSIONES

2.8.1. BENEFICIOS

- Sistema robusto y profesional con SQL Server.
- Fácil integración en entornos Windows empresariales.

2.8.2. INCONVENIENTES

- Dependencia de Windows y SQL Server.
- Curva de aprendizaje para administrar la base de datos.

3. ANÁLISIS

3.1. INTRODUCCIÓN

Este apartado detalla el análisis del sistema de gestión de fichajes, incluyendo requisitos funcionales y no funcionales, diagramas UML y el modelo de datos. Se ha realizado un estudio exhaustivo de las entidades y sus relaciones para garantizar que el sistema cumpla con las necesidades de la empresa y los trabajadores.

3.2. REQUISITOS FUNCIONALES DE USUARIOS

Usuario: Empleado

- RF-01: Iniciar sesión con credenciales (usuario/contraseña).
- RF-02: Registrar entrada/salida con marca de tiempo y método de fichaje (tarjeta, huella, etc.).
- RF-03: Consultar su historial de fichajes y horarios asignados.
- RF-04: Solicitar vacaciones o permisos.
- RF-05: Reportar incidencias (retrasos, ausencias).

Usuario: Administrador (RRHH)

- RF-06: Gestionar altas, bajas y modificaciones de empleados.

- RF-07: Asignar horarios y departamentos.
- RF-08: Aprobar/rechazar solicitudes de vacaciones o permisos.
- RF-09: Generar informes de asistencia, incidencias y horas extras.
- RF-10: Configurar métodos de fichaje (ej: habilitar huella digital).

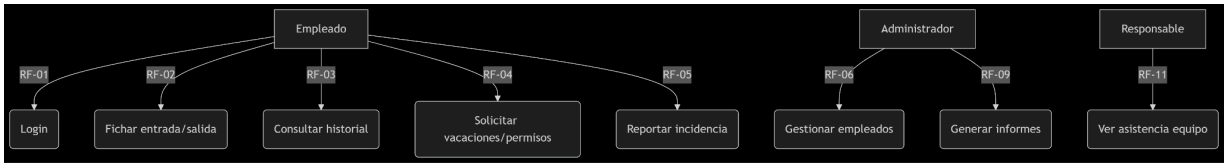
Usuario: Responsable de Departamento

- RF-11: Visualizar asistencia de su equipo.
- RF-12: Validar incidencias reportadas por empleados.

3.3. REQUISITOS NO FUNCIONALES

- RNF-01: Compatibilidad con Windows 10/11 y SQL Server 2019+.
- RNF-02: Tiempo de respuesta < 2 segundos en operaciones CRUD.
- RNF-03: Encriptación de datos sensibles (contraseñas, DNI).
- RNF-04: Interfaz intuitiva (máximo 3 clics para fichar).
- RNF-05: Backup automático diario de la base de datos.

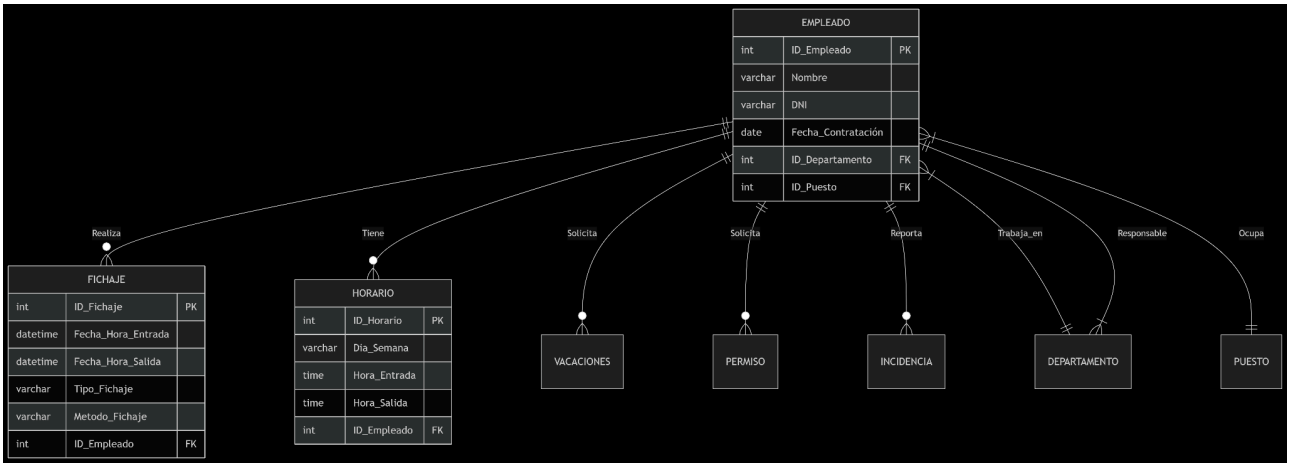
3.4. DIAGRAMAS DE CASOS DE USO/ CASOS DE USO



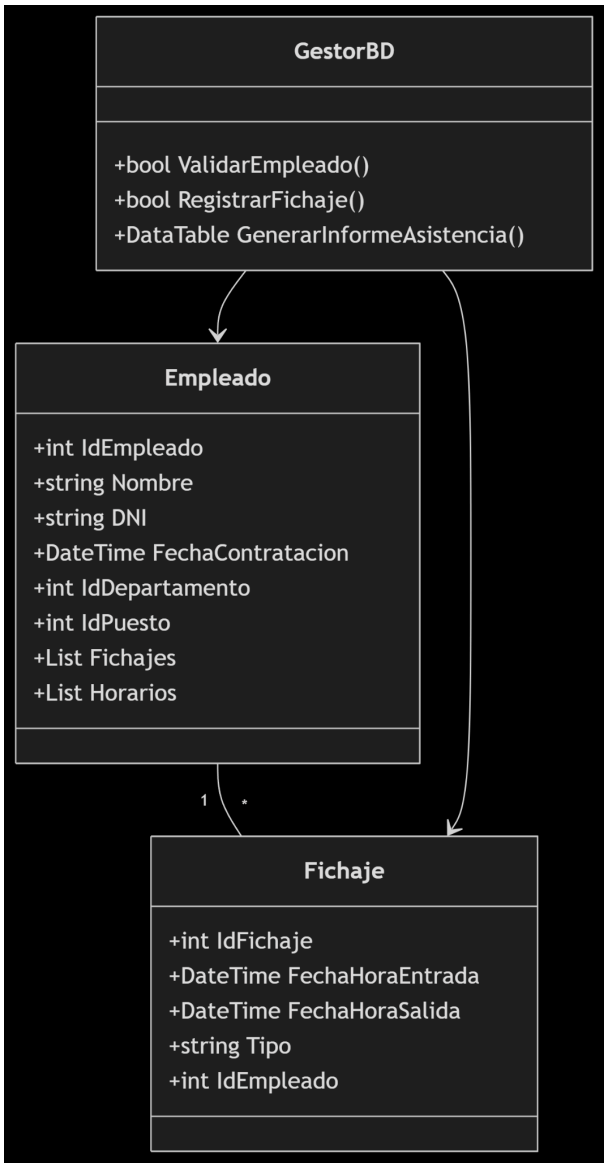
Caso de uso destacado: "Registrar fichaje"

- Precondición: Empleado autenticado.
- Flujo básico:
  1. El Empleado selecciona "Fichar entrada".
  2. El sistema valida el método de fichaje (ej: huella).
  3. Se registra Fecha_Hora_Entrada y Ubicación Fichaje en la tabla FICHAJES.
- Flujo alternativo:
  - Si el empleado está fuera del horario asignado (HORARIO), se marca como "Retraso".

3.5. DIAGRAMAS LÓGICO DE DATOS

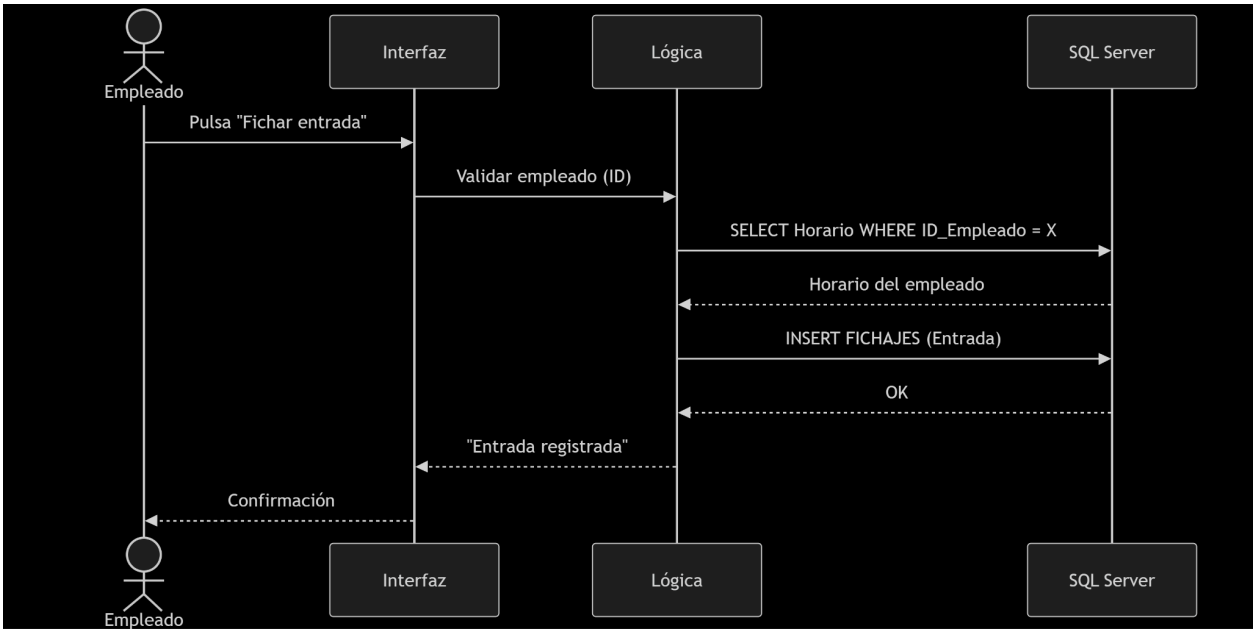


3.6. DIAGRAMAS DE CLASES



3.7. DIAGRAMAS DE INTERACCIÓN

Secuencia: "Registrar fichaje"



3.8. CONCLUSIÓN DE ANÁLISIS

- Cobertura de requisitos: El sistema satisface las necesidades de empleados, administradores y responsables.
- Modelo de datos robusto: Las entidades y relaciones cubren todos los escenarios (fichajes, horarios, incidencias).
- Tecnologías adecuadas: C# y SQL Server garantizan escalabilidad y seguridad.

4. DISEÑO

4.1. INTRODUCCIÓN

En esta sección se define la arquitectura técnica del sistema, las tecnologías seleccionadas y la estructura de la base de datos. El diseño se alinea con los requisitos analizados previamente y garantiza escalabilidad, seguridad y facilidad de mantenimiento.

4.1.1. SELECCIÓN DEL ENTORNO DE DESARROLLO

- Lenguaje de programación: C# (por su integración con Windows y SQL Server).
- Entorno de desarrollo: Visual Studio 2022 (Community Edition, gratuita).
    - Ventajas:
        - Soporte para .NET Framework y .NET Core.
        - Herramientas integradas para diseño de interfaces (Windows Forms/WPF).
        - Depurador avanzado y conectividad con SQL Server.
- Control de versiones: Git + GitHub (para gestión del código fuente).

4.1.2. SELECCIÓN DE BASE DE DATOS

- Motor de base de datos: Microsoft SQL Server 2019 (Express Edition).
  - Ventajas:
    - Compatibilidad nativa con C# (Entity Framework).
    - Soporte para transacciones ACID y encriptación de datos.
    - Licencia gratuita para desarrollo (SQL Server Express).
- Alternativas consideradas y descartadas:
  - MySQL: Menor integración con el ecosistema Microsoft.
  - SQLite: No soporta múltiples usuarios concurrentes eficientemente.

4.2. CONFIGURACIÓN DE LA PLATAFORMA

- Requisitos mínimos:
  - Hardware:
    - Procesador x64 de 2 GHz.
    - 4 GB de RAM.
    - 10 GB de almacenamiento (para SQL Server).
  - Software:
    - Windows 10/11.
    - .NET Framework 4.8 o .NET 6.0.
- Configuración inicial:
  - Instalar Visual Studio con los workloads:
    - ".NET desktop development".
    - "Data storage and processing" (para SQL Server).
  - Instalar SQL Server Management Studio (SSMS) para administrar la base de datos.

4.3. CAPAS DE LA APLICACIÓN

Capa	Tecnologías	Responsabilidad
Presentación	**Windows Forms.NET**	**Interfaz de usuario (formularios de fichaje, login, informes).**
Lógica	**C# + Entity Framework**	**Validación de reglas de negocio (ej: cálculo de horas extras).**
Datos	**SQL Server + Stored Procedures**	**Almacenamiento y recuperación de datos.**

4.4. ESTRUCTURA DE LA BASE DE DATOS

Tablas de la base de datos:

- **EMPLEADO:**

```
CREATE TABLE EMPLEADO (
    ID_Empleado INT IDENTITY(1,1) PRIMARY KEY,
    Nombre NVARCHAR(50) NOT NULL,
    Apellido1 NVARCHAR(100) NOT NULL,
    Apellido2 NVARCHAR(100) NOT NULL,
    DNI_NIF NVARCHAR(20) NOT NULL UNIQUE,
    Fecha_Nacimiento DATE,
    Direccion NVARCHAR(200),
    Telefono NVARCHAR(20),
    Email NVARCHAR(100),
    Contrasenia NVARCHAR(8),
    Fecha_Contratacion DATE NOT NULL,
    ID_Departamento INT NULL,
    ID_Puesto INT NOT NULL
);
```

- **FICHAJE:**

```
CREATE TABLE FICHAJE (
    ID_Fichaje INT IDENTITY(1,1) PRIMARY KEY,
    ID_Empleado INT NOT NULL,
    Fecha_Hora_Entrada DATETIME NOT NULL,
    Fecha_Hora_Salida DATETIME NULL,
    Tipo_Fichaje NVARCHAR(10) NOT NULL CHECK
(Tipo_Fichaje IN ('Entrada', 'Salida')),
    Metodo_Fichaje NVARCHAR(10) NOT NULL CHECK
(Metodo_Fichaje IN ('Tarjeta', 'Huella', 'Facial', 'Web', 'App')),
    Ubicacion_Fichaje NVARCHAR(100),
    Estado NVARCHAR(20) NOT NULL CHECK (Estado IN
('Normal', 'Retraso', 'Ausencia', 'Horas Extra', 'Conflicto')),
    CONSTRAINT FK_Fichaje_Empleado FOREIGN KEY
(ID_Empleado) REFERENCES EMPLEADO(ID_Empleado)
```

```
);
```

- **DEPARTAMENTO:**

```
CREATE TABLE [dbo].[DEPARTAMENTO](
   [ID_Departamento] [int] IDENTITY(1,1) NOT NULL,
   [Nombre] [nvarchar](50) NOT NULL,
   [Descripcion] [nvarchar](200) NULL,
   [ID_Responsable] [int] NULL,
PRIMARY KEY CLUSTERED
(
   [ID_Departamento] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

- **PUESTO:**

```
CREATE TABLE [dbo].[PUESTO](
   [ID_Puesto] [int] IDENTITY(1,1) NOT NULL,
   [Nombre_Puesto] [nvarchar](50) NOT NULL,
   [Descripcion] [nvarchar](200) NULL,
   [Salario_Base] [decimal](10, 2) NOT NULL,
PRIMARY KEY CLUSTERED
(
   [ID_Puesto] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

- **INCIDENCIAS:**

```
CREATE TABLE [dbo].[INCIDENCIA](
    [ID_Incidencia] [int] IDENTITY(1,1) NOT NULL,
   [ID_Empleado] [int] NOT NULL,
   [Tipo] [nvarchar](20) NOT NULL,
   [Descripcion] [nvarchar](500) NOT NULL,
   [Fecha] [date] NOT NULL,
   [Resuelta] [bit] NULL,
PRIMARY KEY CLUSTERED
(
   [ID_Incidencia] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

- **AUSENCIAS:**

```
CREATE TABLE [dbo].[AUSENCIA](
   [ID_Ausencia] [int] IDENTITY(1,1) NOT NULL,
   [ID_Empleado] [int] NOT NULL,
   [Tipo] [nvarchar](20) NOT NULL,
   [Fecha_Inicio] [date] NOT NULL,
   [Fecha_Fin] [date] NULL,
   [Certificado] [bit] NULL,
   [Observaciones] [nvarchar](500) NULL,
PRIMARY KEY CLUSTERED
(
   [ID_Ausencia] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

- **HISTORIAL_PUESTO:**

```
CREATE TABLE [dbo].[HISTORIAL_PUESTO](
   [ID_Historial] [int] IDENTITY(1,1) NOT NULL,
   [ID_Empleado] [int] NOT NULL,
   [ID_Puesto_Anterior] [int] NULL,
   [ID_Puesto_Nuevo] [int] NOT NULL,
   [Fecha_Cambio] [date] NOT NULL,
   [Motivo] [nvarchar](200) NULL,
PRIMARY KEY CLUSTERED
(
   [ID_Historial] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

- **HORARIO:**

```
CREATE TABLE [dbo].[HORARIO](
    [ID_Horario] [int] IDENTITY(1,1) NOT NULL,
    [ID_Empleado] [int] NOT NULL,
    [Dia_Semana] [nvarchar](10) NOT NULL,
    [Hora_Entrada] [time](7) NOT NULL,
    [Hora_Salida] [time](7) NOT NULL,
    [Flexible] [bit] NULL,
PRIMARY KEY CLUSTERED
(
    [ID_Horario] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY],
 CONSTRAINT [UQ_Horario_Empleado_Dia] UNIQUE
NONCLUSTERED
(
    [ID_Empleado] ASC,
    [Dia_Semana] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
```

```
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
```

- **PERMISO:**

```
CREATE TABLE [dbo].[PERMISO](
    [ID_Permiso] [int] IDENTITY(1,1) NOT NULL,
    [ID_Empleado] [int] NOT NULL,
    [Fecha] [date] NOT NULL,
    [Horas] [decimal](4, 2) NOT NULL,
    [Motivo] [nvarchar](200) NOT NULL,
    [Estado] [nvarchar](10) NOT NULL,
PRIMARY KEY CLUSTERED
(
    [ID_Permiso] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
```

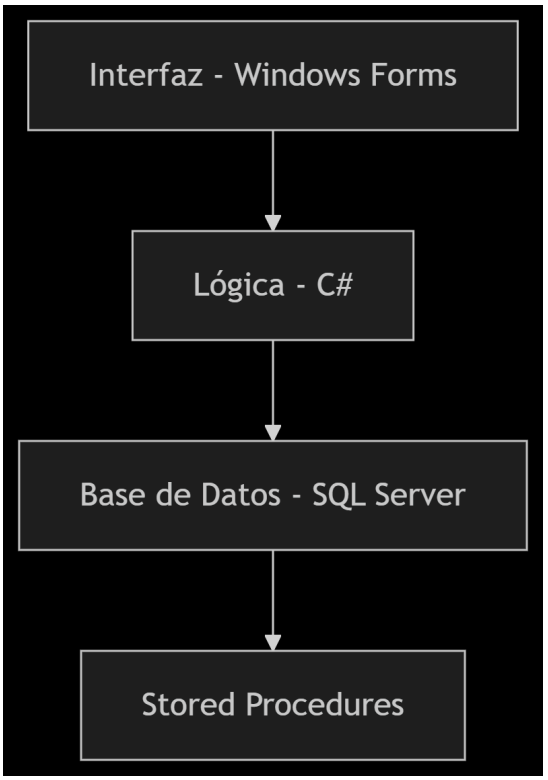
- **VACACIONES:**

```
CREATE TABLE [dbo].[VACACIONES](
    [ID_Vacaciones] [int] IDENTITY(1,1) NOT NULL,
    [ID_Empleado] [int] NOT NULL,
    [Fecha_Inicio] [date] NOT NULL,
    [Fecha_Fin] [date] NOT NULL,
    [Estado] [nvarchar](10) NOT NULL,
    [Motivo] [nvarchar](200) NULL,
PRIMARY KEY CLUSTERED
(
    [ID_Vacaciones] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Relaciones:

- Claves foráneas según el modelo E-R proporcionado.
- Índices en campos de consulta frecuente (ej: ID_Empleado en FICHAJE)

4.5. ARQUITECTURA DE LA APLICACIÓN



Flujo:

- El usuario interactúa con un formulario de Windows Forms.
- La capa de lógica (C#) valida los datos y ejecuta consultas mediante Entity Framework.
- SQL Server procesa las consultas y devuelve resultados

5. IMPLEMENTACIÓN

5.1. INTRODUCCIÓN.

En este apartado indico como se ha realizado la implementación de la estructura de la aplicación.

5.2. CODIFICACIÓN DE LAS DIFERENTES CAPAS

Durante el desarrollo del código se ha utilizado en la medida de lo posible comentarios en el mismo para facilitar la comprensión de lo realizado en caso de necesitar modificar algo. Para el uso de monitorización se ha usado el propio registro del equipo para leer la entrada de caracteres y movimientos del ratón para determinar que estamos en el puesto de trabajo..

5.3. INTEGRACIÓN DE LAS HERRAMIENTAS DE APOYO

Como medida de apoyo usamos:

- **SQL SERVER MANAGEMENT STUDIO 20**. con el cual podremos crear la base de datos y obtener las sentencias fácilmente, además de visualizar la base de datos.
- **GITHUB**. Como gestor de versiones

6. PRUEBAS

6.1. INTRODUCCIÓN

Las pruebas serán realizadas manualmente, tanto por el desarrollador durante la codificación de la aplicación, como por los trabajadores que colaborarán como testers de las distintas versiones de la app.

6.2. PRUEBAS

Pruebas de la Pantalla Principal

Carga inicial:

- Verificar que la pantalla carga correctamente al iniciar la aplicación.
- Comprobar que muestra el mensaje de bienvenida y la fecha actual.

Funcionalidad de fichaje:

- Probar que los botones "Fichar Entrada" y "Fichar Salida" responden correctamente.
- Verificar que se registra la hora exacta del fichaje en la base de datos.
- Comprobar que se muestra un mensaje de confirmación al realizar el fichaje.

Visualización de datos:

- Confirmar que la información del empleado (nombre, departamento) se muestra correctamente.
- Verificar que los botones del menú están distribuidos adecuadamente sin solapamientos.
- Pruebas de la Pantalla de Administración

Gestión de empleados:

- Probar que se pueden añadir nuevos empleados con todos los campos requeridos.
- Verificar que se muestran mensajes de error si faltan campos obligatorios.
- Comprobar que la lista de empleados se actualiza inmediatamente después de añadir o eliminar uno.

Asignación de horarios:

- Verificar que se pueden asignar horarios a los empleados.
- Comprobar que el sistema valida que las horas de entrada sean anteriores a las de salida.
- Pruebas de la Pantalla de Informes

Generación de informes:

- Probar que se pueden generar informes diarios, semanales y mensuales.
- Verificar que los informes incluyen todos los datos relevantes (horas trabajadas, retrasos, ausencias).

Exportación de datos:

- Comprobar que los informes se pueden exportar a formato PDF y Excel.
- Verificar que los archivos exportados contienen la información correcta.
- Pruebas de la Pantalla de Configuración

Ajustes del sistema:

- Probar que se pueden cambiar las preferencias (ej: formato de hora, idioma).
- Verificar que los cambios se guardan correctamente y persisten al reiniciar la aplicación.

Respaldo de datos:

- Comprobar que la función de respaldo genera un archivo correctamente.
- Verificar que se puede restaurar la base de datos desde un respaldo.
- Pruebas Adicionales

Rendimiento:

- Medir el tiempo de respuesta al cargar listas con muchos registros.
- Verificar que la aplicación no se bloquea durante operaciones largas.

Seguridad:

- Probar que las contraseñas se almacenan cifradas.
- Verificar que se requieren permisos adecuados para funciones administrativas.

Usabilidad:

- Confirmar que los mensajes de error son claros y ayudan al usuario a resolver problemas.
- Verificar que la navegación entre pantallas es intuitiva.

6.3. RESULTADOS OBTENIDOS

En las pruebas del desarrollador hemos visto varios fallos, como eran durante el desarrollo se han resuelto sobre la marcha antes de seguir. A los testers se les entregó versiones funcionales, solo buscábamos errores que no hubiésemos encontrado antes. Las pruebas de las personas que llegaron a probarlo transcurrieron sin problema**s.**

6.4. CONCLUSIONES

En general las pruebas han sido satisfactorias, aunque esto no quita de que puedan surgir nuevos problemas que haya que corregir

7. CONCLUSIONES

7.1. CONCLUSIONES FINALES

El desarrollo de la app ha sido largo y complicado, y aunque aún no esté terminada ha sido muy gratificante ya que la idea de crear esta app es puramente personal y facilitará, en mi opinión, la planificación de sus jornadas a los pescadores deportivos, que como yo actualmente necesitamos usar varias aplicaciones y páginas web. Como explico, la aplicación no está acabada, aunque creo que en el reducido tiempo que he tenido para realizarla he tratado todos los temas y tecnologías que usaré

7.2. DESVIACIONES TEMPORALES

He tenido algunas desviaciones temporales de gran impacto, como por ejemplo usar correctamente la cámara, ya que la información de como usarla está desactualizada o no completa y me ha quitado mucho tiempo (al menos una semana completa). También se ha perdido mucho tiempo en usar correctamente las APIs climatológicas y el GPS. Al intentar subir la app a Google Play y como esta hace uso de la cámara, se me requiere una política de privacidad que debe estar alojada en una web, por lo que he tenido que crear una política de privacidad, una web y subirla a un hosting, lo que ha retrasado la subida de la app a Google Play.

7.3. POSIBLES AMPLIACIONES Y MODIFICACIONES

En un futuro, fuera del proyecto, voy a seguir terminando la app y pienso añadir las siguientes funcionalidades:

- Uso de Mapas
- Integración de Previsión, Vientos y Mareas
- Uso de una Base de Datos en un servidor
- Implementación de la Galería de Fotos
- Compartir Registros
- Uso de control por voz para añadir registros

7.4. VALORACION PERSONAL

Pienso que la aplicación está bastante bien para el tiempo que he podido dedicarle y ayudará mucho a todo el pescador que decida usarla. Pero sobre todo me ha dado una gran satisfacción realizarla ya que era una aplicación que yo y mis amigos necesitábamos para organizar nuestra afición.

8. BIBLIOGRAFÍA

Referencias Técnicas

- Microsoft Documentation (2023). Documentación oficial de SQL Server.

  Disponible en: https://docs.microsoft.com/en-us/sql/sql-server/

- Microsoft Learn (2023). Guía de C# y .NET Framework.

  Disponible en: https://learn.microsoft.com/en-us/dotnet/csharp/

- Entity Framework Core Documentation (2023). Entity Framework Core - Getting Started.

  Disponible en: https://docs.microsoft.com/en-us/ef/core/

- Newtonsoft.Json (2023). Documentación oficial para manejo de JSON en C#.

  Disponible en: https://www.newtonsoft.com/json

Referencias de Diseño y Metodología

- Pressman, R. (2010). Ingeniería del Software: Un enfoque práctico. McGraw-Hill.

  (Para metodologías de desarrollo)

- Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.

  (Para patrones de diseño MVC y Repository)

Normativas Legales

- BOE (2018). *Ley Orgánica 3/2018 de Protección de Datos Personales*.

  Disponible en: https://www.boe.es/eli/es/lo/2018/12/05/3

9. GLOSARIO

Término	Definición
C#	Lenguaje de programación orientado a objetos desarrollado por Microsoft.
SQL Server	Sistema de gestión de bases de datos relacionales de Microsoft.
Entity Framework	ORM (Mapeo Objeto-Relacional) para .NET que simplifica el acceso a bases de datos.
MVC	Patrón de diseño Modelo-Vista-Controlador para separar lógica, interfaz y datos.
Stored Procedure	Conjunto de comandos SQL almacenados en el servidor de base de datos.
DTO (Data Transfer Object)	Objeto que transporta datos entre procesos.
RGPD	Reglamento General de Protección de Datos de la UE.