

Técnico Superior en DAM – DAW - ASIR

Título del Trabajo Fin de Estudios

Trabajo fin de estudio presentado por:	Ana González Bueno Daniel Torres Retuerto
Tipo de trabajo:	TFC
Tutor/a:	Damián Sualdea
Fecha:	

Resumen

Presentamos una solución de control horario. Desarrollamos una aplicación que cumpliendo con la normativa laboral, permita el registro horario y el control y gestión de turnos de trabajo.

Ha de permitir que los empleados fichen sin geolocalización ni biometrías para asegurar cumplir la normativa en materia de protección de datos personales. Lo que implica la generación de sistema de acceso mediante claves para cada usuario y proponemos el uso en las instalaciones de trabajo para evitar la geolocalización.

Los datos de los empleados, han de ser accesibles para cada trabajador de forma personal y para la administración competente.

Los turnos de trabajo, han de poder gestionarse cumpliendo la normativa aplicable en cuanto a cómputo anual de horas trabajadas, horas extras de cualquier tipo, y horas complementarias.

Ha de asegurarse el personal mínimo requerido por la empresa para el desempeño de su actividad.

Palabras clave: turnos de trabajo, control horario, seguridad y accesibilidad.

Índice de contenidos

1. Introducción.....	9
1.1. Justificación.....	9
1.2. Objetivos.....	9
2. Módulos formativos aplicados en el trabajo.....	10
3. Herramientas y lenguajes utilizados.....	11
4. Metodologías utilizadas.....	12
5. Componentes del equipo y aportaciones realizadas por cada alumno.....	13
5.1. Estudio de mercado.....	13
5.1.1. Tabla comparativa de aplicaciones actualmente en el mercado.....	13
5.1.2. Análisis DAFO de nuestra solución.....	14
5.1.3. Subapartado 1.2.....	14
5.2. Modelo de datos.....	15
5.3. Diagramas UML.....	15
5.3.1. Diagrama de clases.....	15
5.3.2. Clasificación de usuarios.....	16
5.3.3. Caso de uso 01.....	16
5.4. Diseño de interfaces.....	17
5.4.1. Wireframes.....	17
5.4.2. Prototipo de interfaz de alta definición.....	18
5.4.3. Paleta de colores.....	18
5.4.4. Logotipo.....	18
5.5. Planificación temporal y trabajo en equipo.....	19
5.5.1. Presupuesto temporal de tareas.....	19
5.5.2. Organización de tareas y tiempos finales.....	20
5.5.3. Trabajo en equipo.....	21

6.	Conclusiones	22
6.1.	Análisis de desviaciones temporales y de tareas	22
6.2.	Conclusiones generales del proyecto	22
6.2.1.	Evaluación global del proyecto	22
6.2.2.	Reflexión sobre el proceso de aprendizaje y desarrollo	22
6.2.3.	Recomendaciones para futuros proyectos similares	22
6.3.	Limitaciones y prospectiva	22
6.3.1.	Posibles mejoras y ampliaciones del proyecto	22
6.3.2.	Nuevas líneas de investigación o desarrollo que podrían derivarse del proyecto.22	
6.3.3.	Sugerencias para la implementación en entornos reales	22
7.	Referencias bibliográficas	23
Anexo A.	Diagramas de GANTT	24
Anexo B.	Código fuente de la solución y pruebas	25
Anexo C.	Manual de instalación - despliegue	26
Anexo D.	Documentación de la API	27
Anexo E.	Otros anexos de interés	28

Índice de figuras

Tabla 1: Herramientas, lenguajes, frameworks y APIs utilizadas	10
Tabla 2 Presupuesto temporal de tareas	11
Tabla 3 Comparativa de aplicaciones actualmente en el mercado.....	13
Ilustración 1 Análisis DAFO.....	14
Ilustración 2 Diagrama E/R.....	15
Ilustración 3 Diagrama de clases.....	16
Ilustración 4 Caso de uso "Recepción de pedido"	16
Ilustración 5 Wireframes.....	17
Ilustración 6 Prototipo de interfaz de alta definición.....	17
Ilustración 7 Paleta de colores	18
Ilustración 8 Logotipo en positivo.....	18
Ilustración 9 Logotipo en negativo.....	19

Índice de tablas

Tabla 1: Herramientas, lenguajes, frameworks y APIs utilizadas	10
Tabla 2 Presupuesto temporal de tareas	11
Tabla 3 Comparativa de aplicaciones actualmente en el mercado.....	13

1. Introducción

1.1. Justificación

Desarrollo de una aplicación de fácil manejo para el registro horario, y la gestión de turnos de trabajo (solicitud de cambios y gestión de los mismos) de forma sencilla y semiautomática.

1.2. Objetivos

Automatizar cambios de turno y horarios asegurando el cumplimiento de la normativa general aplicable, dejando lugar pequeños ajustes para adaptarlo a cada convenio laboral que aplique cambios de mejora respecto del general.

- Objetivo general : automatizar cambios de turno.
- Objetivos específicos : ampliar a la necesidad legal de control horario y asegurar cumplimiento de requisitos horarios fijados por cada convenio colectivo de forma automática.

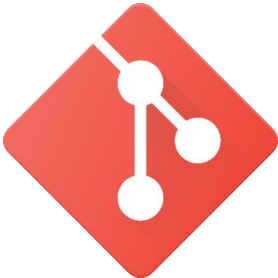
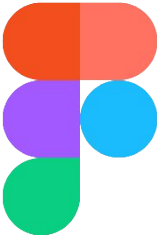
2. Módulos formativos aplicados en el trabajo

Identifica los módulos formativos y resultados de aprendizaje aplicados en el presente trabajo (consultar legislación – puedes pedir ayuda a tu tutor)



3. Herramientas y lenguajes utilizados

Especifica los lenguajes, frameworks, APIs y herramientas utilizadas, junto con una breve descripción de estas.

Tabla 1: Herramientas, lenguajes, frameworks y APIs utilizadas

<p><u>Git</u></p> 	<p>Git es un sistema de control de versiones distribuido, creado por Linus Torvalds en 2005. Su propósito principal es gestionar el desarrollo del kernel de Linux, pero su flexibilidad y eficiencia lo han convertido en una herramienta ampliamente adoptada en la industria del software. Git permite a los desarrolladores rastrear cambios en el código fuente, colaborar en proyectos y revertir a versiones anteriores si es necesario. Utiliza un modelo de datos basado en instantáneas, lo que garantiza la integridad y consistencia de los datos. Además, Git facilita la creación de ramas y fusiones, lo que permite a los equipos trabajar en paralelo sin conflictos (Kranio, 2023).</p>
<p><u>Figma</u></p> 	<p>Figma es una herramienta de diseño de interfaces basada en la web que permite a los usuarios trabajar de manera colaborativa en tiempo real desde cualquier dispositivo con acceso a Internet. Ofrece un espacio de trabajo flexible para crear diseños y prototipos de manera eficiente, facilitando la colaboración y la consistencia en los proyectos.</p>

<p><u>Angular</u></p> 	<p>Angular es un framework de desarrollo web basado en TypeScript, desarrollado y mantenido por Google. Se utiliza para crear aplicaciones de una sola página (SPA) con una arquitectura modular y basada en componentes. Ofrece herramientas como el enlace de datos bidireccional, inyección de dependencias y una estructura escalable que facilita el mantenimiento y desarrollo de aplicaciones complejas.</p>
<p><u>Spring Tools</u></p> 	<p>Spring Tools Suite (STS) es un entorno de desarrollo basado en Eclipse, diseñado específicamente para aplicaciones Java con Spring Framework. Incluye herramientas que facilitan la configuración, desarrollo y depuración de aplicaciones Spring Boot, proporcionando soporte para integraciones con bases de datos, servidores y microservicios.</p>
<p><u>MySQL Workbench</u></p> 	<p>MySQL Workbench es una herramienta de diseño y administración para bases de datos MySQL. Permite la modelación de bases de datos, ejecución de consultas SQL, administración de servidores y optimización de rendimiento. Es ampliamente utilizado en el desarrollo de aplicaciones que requieren almacenamiento y gestión de datos estructurados.</p>

<p><u>TypeScript</u></p> 	<p>TypeScript es un lenguaje de programación desarrollado por Microsoft que extiende JavaScript con tipado estático opcional. Es utilizado en el desarrollo de aplicaciones modernas, especialmente con frameworks como Angular, permitiendo un código más estructurado y mantenible.</p>
<p><u>Java</u></p> 	<p>Java es un lenguaje de programación orientado a objetos y multiplataforma, ampliamente utilizado en el desarrollo de aplicaciones empresariales, web y móviles. Su compatibilidad con el ecosistema de Spring Boot lo convierte en una opción ideal para el desarrollo de aplicaciones backend robustas y escalables.</p>
<p><u>MySQL</u></p> 	<p>MySQL es un sistema de gestión de bases de datos relacional de código abierto, ampliamente utilizado en aplicaciones web y empresariales. Ofrece alto rendimiento, escalabilidad y seguridad, siendo una de las opciones más populares para almacenar y gestionar grandes volúmenes de datos.</p>
<p><u>Bootstrap</u></p> 	<p>Bootstrap es un framework de diseño web basado en HTML, CSS y JavaScript, utilizado para la creación de interfaces responsivas y atractivas. Proporciona una amplia variedad de componentes predefinidos como botones, formularios, alertas y barras de navegación, facilitando el desarrollo de aplicaciones con un diseño moderno y consistente.</p>

Swagger UI



Swagger UI es una herramienta de código abierto que permite visualizar y probar APIs REST de manera interactiva. Forma parte del ecosistema Swagger y facilita la documentación de servicios web, permitiendo a los desarrolladores explorar endpoints, enviar peticiones y comprender la estructura de las APIs sin necesidad de herramientas adicionales. Es ampliamente utilizado en aplicaciones que exponen servicios RESTful con Spring Boot.

4. Metodologías utilizadas

Describe la metodología de desarrollo de proyecto aplicada

5. Componentes del equipo y aportaciones realizadas por cada alumno

Aquí debéis indicar quienes sois Fases del proyecto

5.1. Estudio de mercado

Estudio de la temática a desarrollar, análisis de la competencia y mejoras a realizar, estudio del usuario de la aplicación.

5.1.1. Tabla comparativa de aplicaciones actualmente en el mercado

Tabla 3 Comparativa de aplicaciones actualmente en el mercado

Característica	Bizneo	Clockify	Factorial	FichaWork
Descripción	Software para gestión de tiempo	Software para gestión de tiempo	Software para gestión de tiempo	Software para gestión de tiempo
Funcionalidades	Control horario, gestión de turnos, control de ausencias y vacaciones. Permite integración de más servicios de rrhh	Control horario, gestión de turnos, control de ausencias y vacaciones. Gestión de tiempo, registro actividad e incluso proyectos. Gastos y ganancias	Control horario, gestión de turnos, control de ausencias y vacaciones. Permite integración de más servicios de rrhh.	Control horario, control de ausencias y vacaciones
Público	Pequeña,	Pequeña	Pequeña,	Pequeña

Característica	Bizneo	Clockify	Factorial	FichaWork
Objetivo	mediana y gran empresa	empresa individual e	mediana y gran empresa.	empresa e individual
Plataformas	Mac y Windows, iOS, Android.	iOS, Android.	Mac y Windows, iOS, Android.	
Precio	Empleado/mes	Gratis con compras dentro de la app (suscripcion Pro).	Suscripción anual.	Empleado/mes
Puntuación	4.6/5.0	4.7/5.0	4.7/5.0	4.6/5.0

5.1.2. Análisis DAFO de nuestra solución



Ilustración 1 Análisis DAFO

Describe las diferentes debilidades, amenazas, fortalezas y oportunidades

5.1.3. Subapartado 1.2

Texto Normal del menú de estilos.

5.2. Modelo de datos

Diseño de la base de datos

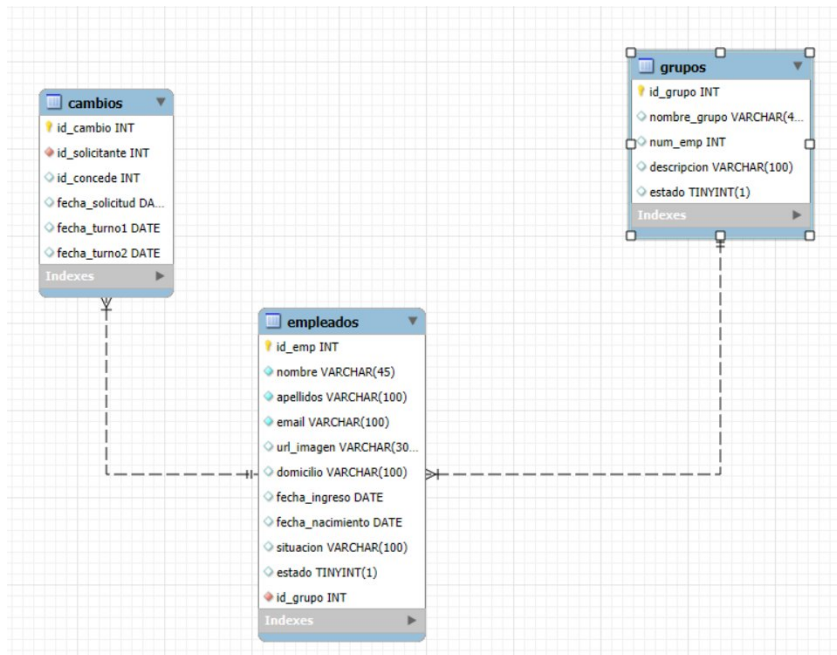


Ilustración 2 Diagrama E/R

5.3. Diagramas UML

El diagrama de clases es un modelo estático que representa la estructura del sistema, mostrando sus clases, atributos, métodos y las relaciones entre ellas. Es una representación fundamental en el diseño orientado a objetos y ayuda a definir cómo los objetos interactúan en el sistema.

El diagrama de casos de uso es un modelo dinámico que representa las interacciones entre los actores (usuarios o sistemas externos) y el sistema. Se utiliza para describir los requisitos funcionales desde la perspectiva del usuario.

5.3.1. Diagrama de clases

Empleado 1 → * Grupo (Un empleado pertenece a un grupo)

Grupo 1 → * Cambio (Un grupo puede tener muchos cambios)

Empleado 1 → * Cambio (Un empleado puede solicitar muchos cambios)

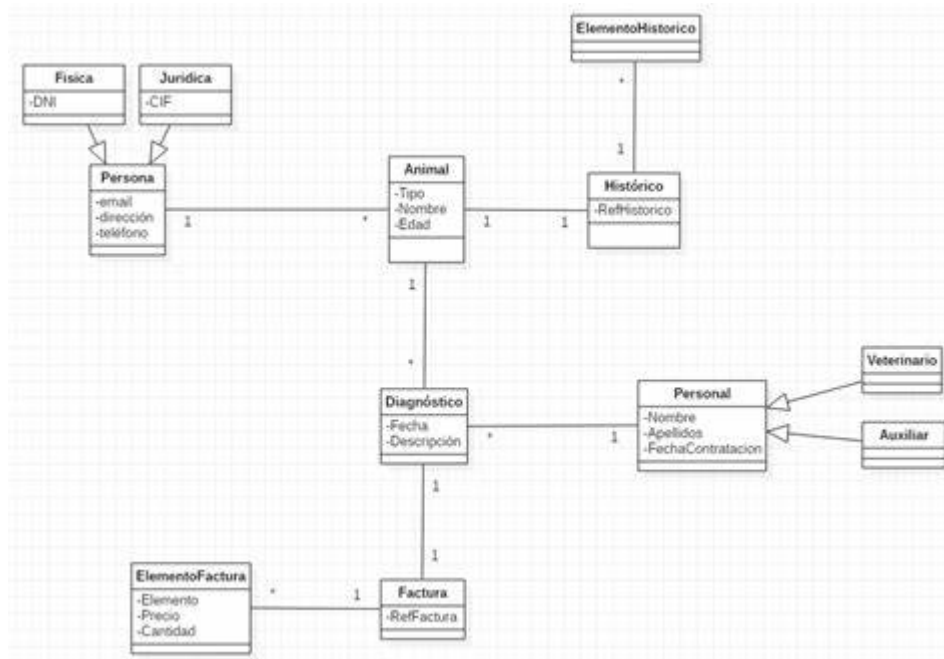


Ilustración 3 Diagrama de clases

5.3.2. Clasificación de usuarios

En el sistema desarrollado, existen diferentes tipos de usuarios con roles y permisos específicos. Para definir estos perfiles, se ha utilizado el método Persona (UX) del Design Thinking Services (2023), que ayuda a representar los usuarios con base en sus necesidades, objetivos y comportamientos.

Tipos de Usuarios y Perfiles

1. Administrador

Usuario encargado de la gestión del sistema. Tiene acceso total para administrar empleados, grupos y realizar modificaciones en la estructura del sistema.

Objetivos y Necesidades

- Gestionar empleados (altas, bajas y modificaciones).
- Asignar empleados a grupos y definir turnos.
- Cambiar el estado de los empleados (activo/inactivo).
- Monitorear la organización de los equipos de trabajo.

Funciones en el sistema

- Acceso completo a la plataforma.
- Creación y edición de empleados y grupos.
- Modificación de la asignación de empleados.
- Control de acceso y permisos.

2. Empleado

Usuario estándar que pertenece a un grupo de trabajo dentro de la empresa. Puede visualizar su información personal y recibir asignaciones de turnos.

Objetivos y Necesidades

- Consultar su grupo y horario de trabajo.
- Acceder a su perfil y estado laboral.
- Notificar cambios en su disponibilidad.

Funciones en el sistema

- Visualización de su información personal.
- Consulta de horarios y grupo asignado.
- Notificación de disponibilidad y estado.

5.3.3. Caso de uso 01

Caso de Uso: Gestión de Empleados y Grupos

Descripción del Caso de Uso

Este caso de uso describe cómo los administradores pueden gestionar los empleados y asignarlos a grupos dentro del sistema. El flujo incluye la verificación del estado del empleado y la asignación de grupos según disponibilidad.

Actores

- **Administrador:** Tiene permisos para gestionar empleados y asignarlos a grupos.
- **Sistema:** Plataforma que permite visualizar y modificar la información de empleados y grupos.

Flujo Principal

1. El **administrador** accede a la sección de gestión de empleados.
2. Selecciona un empleado de la lista.
3. Verifica su estado actual (**activo/inactivo**).
4. Si es necesario, cambia su estado.
5. Asigna o cambia el grupo del empleado.
6. El sistema actualiza la información en la base de datos.

Extensiones

- Si el empleado está **inactivo**, el sistema impide su asignación a un grupo.

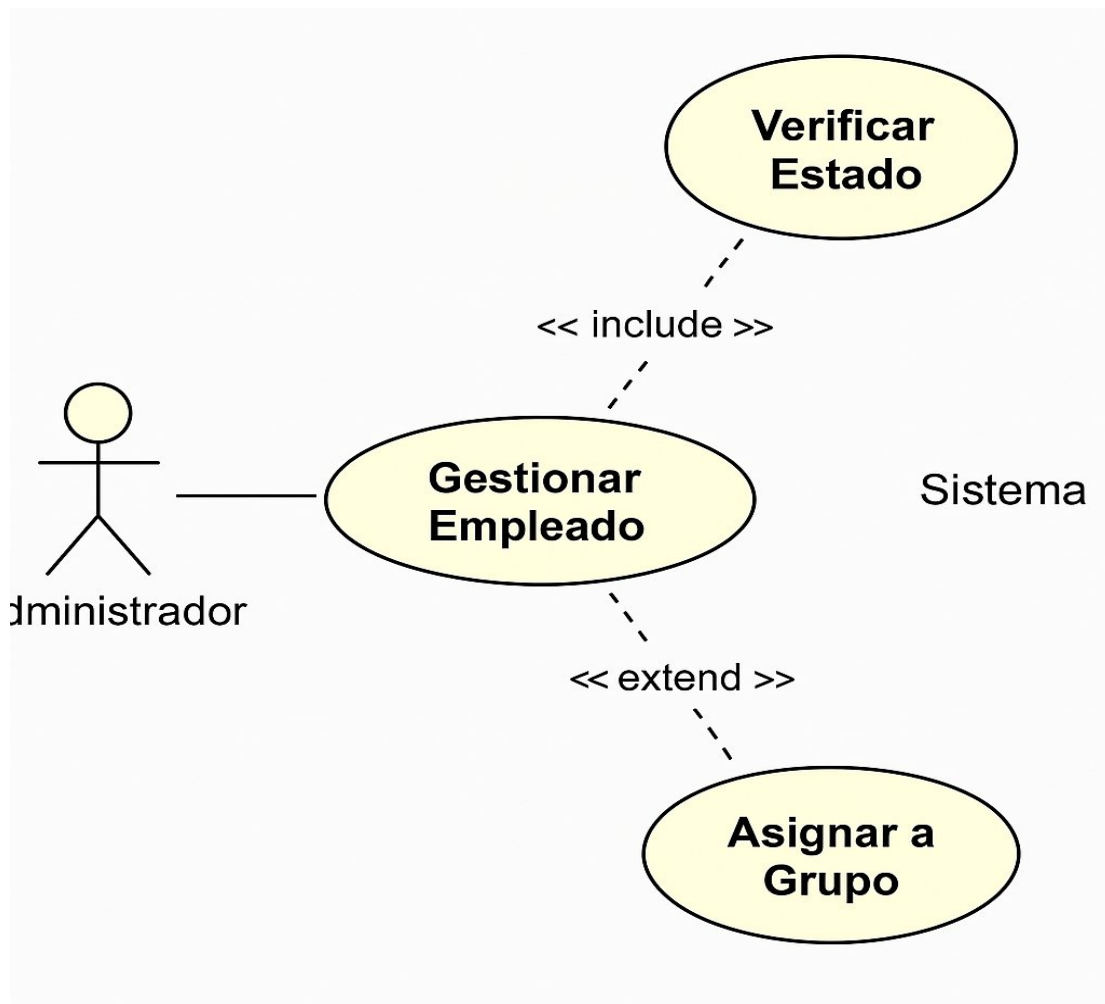


Ilustración 4 Caso de uso "Recepción de pedido"

5.4. Diseño de interfaces

5.4.1. Wireframes

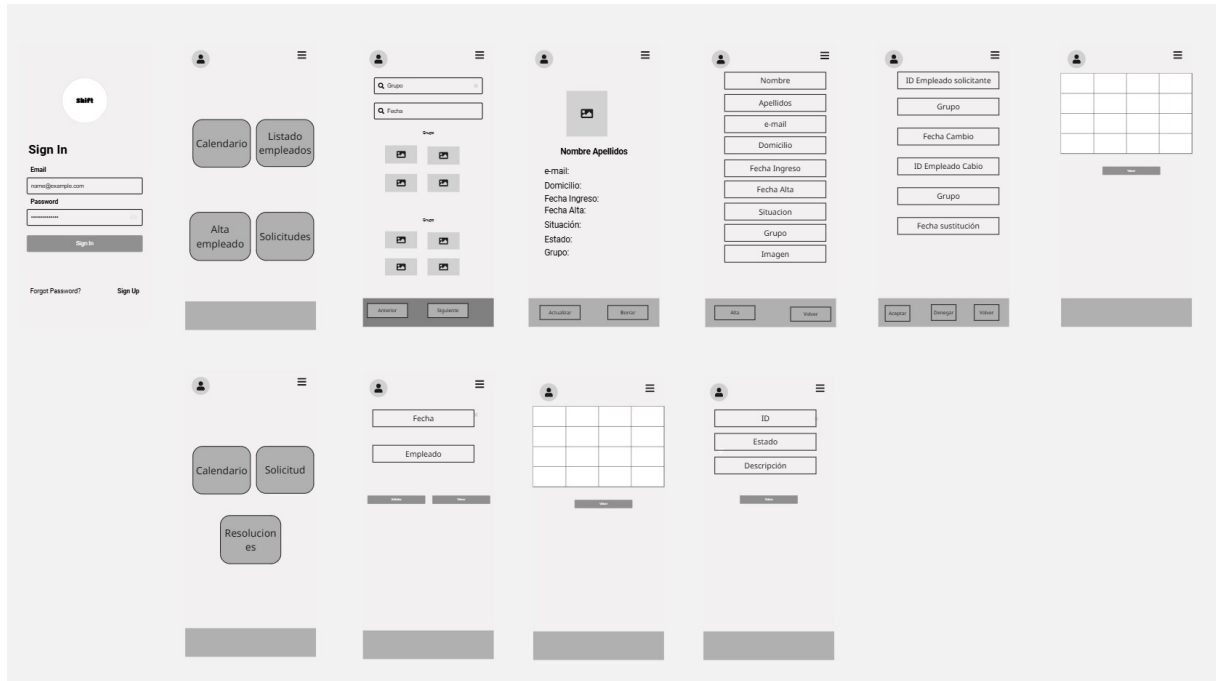


Ilustración 5 Wireframes

5.4.2. Prototipo de interfaz de alta definición

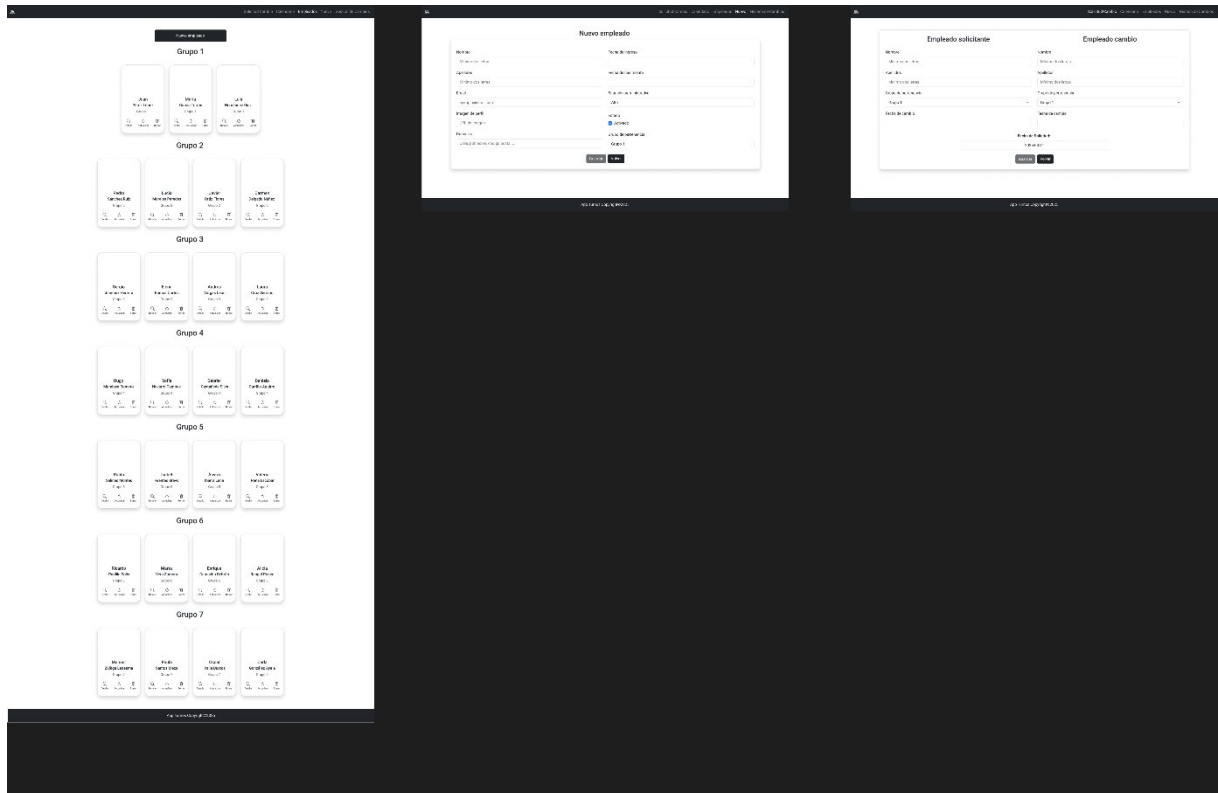


Ilustración 6 Prototipo de interfaz de alta definición

5.4.3. Paleta de colores

Explicación de la elección de colores

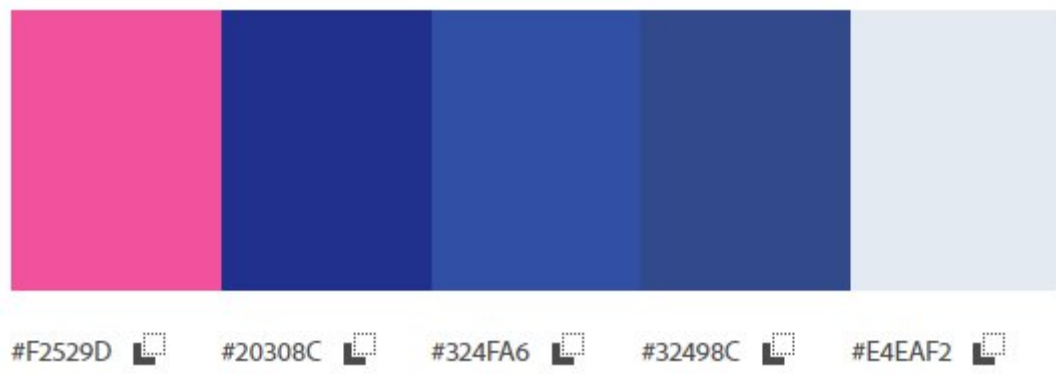


Ilustración 7 Paleta de colores

<https://color.adobe.com/es/create/image>

5.4.4. Logotipo

Explicación de la elección de logotipo



Ilustración 8 Logotipo en positivo

Logotipo en negativo



Ilustración 9 Logotipo en negativo

5.5. Planificación temporal y trabajo en equipo

5.5.1. Presupuesto temporal de tareas

Puedes basarte en la tabla siguiente

Tabla 2 Presupuesto temporal de tareas

Tareas	Subtareas	Duración (horas)	Persona/s asignada/s
Planificación del Proyecto	Definir objetivos y alcance		
	Identificar recursos		
	Crear plan de proyecto		

	...		
Diseño de la Aplicación	Crear wireframes y maquetas		
	Diseñar arquitectura de BD		
	Definir experiencia de UX		
	...		
Desarrollo Frontend	Implementar interfaz de usuario		
	Integrar diseño responsivo		
	Realizar pruebas de usabilidad		
	...		
Desarrollo Backend	Configurar servidor y BD		
	Implementar lógica de negocio		
	Crear APIs y servicios web		
	...		
Integración y Pruebas	Integrar frontend y backend		
	Realizar pruebas funcionales		
	Corregir errores y optimizar		
	...		
Despliegue y Lanzamiento	Configurar entorno de producción		
	Desplegar la aplicación		
	Realizar pruebas finales		
	...		

5.5.2. Organización de tareas y tiempos finales

Puedes basarte en la tabla anterior

En los anexos incorpora el diagrama de Gantt correspondiente al presupuesto y el resultante al final del proyecto.

5.5.3. Trabajo en equipo

Debes definir cuáles han sido vuestras aportaciones.

Se debe aportar, además, el/los gráficos de aportaciones al repositorio de GitHub

6. Conclusiones

6.1. Análisis de desviaciones temporales y de tareas

Realiza un análisis de las diferencias entre el presupuesto inicial de tareas y tiempos y las tareas y tiempos realizados finalmente.

Puedes apoyarte en una tabla comparativa, comentando los motivos de las posibles desviaciones.

6.2. Conclusiones generales del proyecto

6.2.1. Evaluación global del proyecto.

6.2.2. Reflexión sobre el proceso de aprendizaje y desarrollo.

6.2.3. Recomendaciones para futuros proyectos similares.

6.3. Limitaciones y prospectiva

6.3.1. Posibles mejoras y ampliaciones del proyecto.

6.3.2. Nuevas líneas de investigación o desarrollo que podrían derivarse del proyecto.

6.3.3. Sugerencias para la implementación en entornos reales.

7. Referencias bibliográficas

- Design Thinking Services. (2023). *Método persona*. (D. T. Services, Ed.) Recuperado el 01 de 10 de 2024, de <https://www.designthinking.services/herramientas-design-thinking/metodo-persona/>
- Figma. (2023). *Figma: Herramienta de diseño de interfaces*. Recuperado el 20 de 10 de 2024, de <https://www.figma.com>
- Kranio. (29 de 08 de 2023). *Descubriendo Git: Características y Ventajas*. Recuperado el 10 de 2024, de <https://www.kranio.io/blog/descubriendo-git-caracteristicas-y-ventajas>

ANEXO A. DIAGRAMAS DE GANTT

ANEXO B. CÓDIGO FUENTE DE LA SOLUCIÓN Y PRUEBAS

ANEXO C. MANUAL DE INSTALACIÓN - DESPLIEGUE

ANEXO D.DOCUMENTACIÓN DE LA API

En caso de que se implemente una API

ANEXO E. OTROS ANEXOS DE INTERÉS

Puedes crear los que consideres necesarios – de acuerdo con el tutor de proyecto – según la necesidad y naturaleza del proyecto.