Manual técnico

Proyecto final de Grado Superior de Desarrollo de Aplicaciones Multiplataforma

Estudiante:

Héctor Martínez San José

ÍNDICE

1. Descripción del proyecto y justificación

1.1. Introducción al cliente

Stellantis N.V. es un grupo automovilístico internacional ítalo-franco-estadounidense con sede en Países Bajos, que fue fundado el 16 de enero de 2021, fruto de la fusión entre iguales del ítalo-estadounidense Fiat Chrysler Automobiles y el francés Groupe PSA. Resultando en uno de los mayores fabricantes de automóviles del mundo y albergando bajo su nombre 14 marcas de automóvil.

El fabricante cuenta con un total de 33 plantas repartidas por todo el planeta, con especial predominancia de Europa, donde cuenta con casi 20 plantas, así como en Norteamérica, donde cuenta con un total de 10 factorías. Cuenta además con un total de 3 fábricas en América del Sur. A nivel de producción y personal, Stellantis da empleo a más de 300.000 personas en todo el mundo, repartidas por las diferentes plantas de producción y concesionarios.

La distribución y organización de todos los elementos implicados en la producción se ha estandarizado en todas sus plantas para aumentar la eficiencia y control de los procesos. Como ejemplo de ello podemos tomar a la factoría que tiene el fabricante en Figueruelas (Zaragoza) que cuenta con 5.248 empleados. La planta se distribuye en 3 grandes secciones interconectadas, donde cada una de ellas se ocupa de un proceso diferente:

- Prensas: Se ocupan de moldear y soldar los elementos metálicos (chasis, carrocerías, etc.).
- Pintura: Se ocupan de pintar todos los componentes que recibe de prensas.
- Acabado final: Se ocupa del montaje de todos los elementos sobre la carrocería.

De las tres secciones, acabado final es por necesidades operativas la que más personal emplea en su actividad, por ello el grupo emplea una organización por equipo, compuesto por 6 trabajadores (1 coordinador y 5 operarios), donde cada equipo se encarga de unas funciones específicas. Los equipos se distribuyen a lo largo de la línea de producción, de manera que cualquier incidencia que pueda surgirle a un trabajador pueda ser detectada, solventada o mejorada en el futuro.

Siendo conscientes de lo exigentes que son los ambientes en las líneas de producción, el grupo aplica no solo normativas generales de seguridad y salud, sino que forma a sus empleados y suma normativas tan importantes como:

- Se acuña el concepto ‘Kaizen’, como la idea de que todo puede ser mejorado y perfeccionado. Se realizan evaluaciones periódicas, para mejorar tanto la ergonomía de los puestos de trabajo como de los procesos productivos. De esta manera se van mejorando todos los procesos.

- Distribución de la jornada laboral diaria en 5 fracciones, separadas por 2 descansos de 10 minutos y uno de 20 minutos. Evitando periodos largos de actividad intensa.

- Rotación obligatoria de puesto cada fracción de jornada, para evitar sobrecargas asociadas a actividades repetitivas.

1.2. Contexto

En el entorno altamente dinámico de las líneas de producción, la eficiencia y la coordinación son cruciales para garantizar un rendimiento óptimo. En la actualidad, la coordinación de personal en líneas de producción se sigue realizando de manera manual con papel y bolígrafo. Los responsables de un equipo, los coordinadores, realizan en sus últimos minutos de jornada una rotación para el día siguiente. Sin embargo, si su equipo tiene algún tipo de incidencia, como una baja laboral o una nueva incorporación de personal, tiene que readaptar la rotación. Además, si ocurrió una incidencia durante el proceso productivo del día anterior, se revisan las hojas de rotaciones para saber que operario estaba en que puesto y a qué hora.

1.3. Objetivo

Con el objetivo de optimizar la gestión de recursos y mejorar la experiencia laboral de los trabajadores, se va a diseñar y desarrollar una aplicación de escritorio en Java destinado a ayudar a los coordinadores de equipo en la generación automática de rotaciones de trabajadores por los diferentes puestos de su sección.

1.4. Descripción del proyecto

El proyecto se enfoca en proporcionar a los coordinadores de equipo una herramienta eficaz y fácil de usar para gestionar las rotaciones diarias de los trabajadores en líneas de producción. Cada coordinador, responsable de cinco trabajadores, supervisa una sección con cinco puestos de trabajo. Durante una jornada laboral estándar, se llevarán a cabo cinco rotaciones, lo que implica que cada trabajador cambiará de puesto de trabajo en cinco ocasiones a lo largo del día. Es importante destacar que los trabajadores no pueden repetir el mismo puesto en el que estuvieron durante la rotación anterior por motivos de salud laboral.

1.5. Justificación del proyecto

Optimización de Recursos: Automatizar las rotaciones permite una distribución equitativa y eficiente de las habilidades y conocimientos de los trabajadores en los diferentes puestos, maximizando así el rendimiento global de la línea de producción.

Mejora en la Experiencia Laboral: Al evitar la repetición de los mismos puestos de trabajo, se promueve la variedad y el desarrollo de habilidades, contribuyendo a una experiencia laboral más enriquecedora para los trabajadores, además de menos lesiva.

Ahorro de Tiempo: El programa elimina la carga manual de la planificación diaria de rotaciones, permitiendo a los coordinadores centrarse en tareas estratégicas y de mayor valor añadido.

Reducción de Errores: La automatización minimiza la posibilidad de errores humanos en la asignación de rotaciones, garantizando un cumplimiento preciso de las condiciones laborales establecidas.

Adaptabilidad: La flexibilidad del programa permite ajustar fácilmente las rotaciones según las necesidades cambiantes de la producción y la disponibilidad de los trabajadores.

1.6. Estudio de mercado

Teniendo claro los objetivos y el contexto, se ha realizado una búsqueda de posibles herramientas ya desarrolladas, con la finalidad de ofrecer un mejor producto. En dicho estudio se concluyó que no había ninguna herramienta específica que realizará la misma función objetivo, solo se encontraron aplicaciones generalistas de generación de turnos de trabajo.

1.7. Conclusiones previas

En conclusión, el desarrollo de este programa en Java ofrece una solución integral para mejorar la gestión de recursos humanos en líneas de producción, proporcionando a los coordinadores de equipo una herramienta efectiva y adaptable. La implementación de este proyecto no solo beneficia a los coordinadores y trabajadores, sino que también contribuye significativamente a la eficiencia operativa de la empresa en su conjunto.

2. Solución propuesta y requisitos técnicos

2.1. Solución propuesta

El proyecto enfoca la solución del problema de la realización manual de rotaciones, como una digitalización de los procesos necesarios para la obtención de las mismas, a la vez que añade características que optimicen la realización de otras tareas asociadas a la figura del coordinador.
Proponiendo la implantación de una aplicación de escritorio que tenga acceso a la información de los trabajadores a través de la base de datos relacional de la empresa.

Diagrama de la arquitectura del sistema

Las funciones que va a realizar la aplicación son:

1. Login de usuarios.
   - Gestión de usuarios que acceden a la aplicación.
   - Recuperación de contraseña si el usuario no se acuerda.
   - Creación de usuarios, solo los coordinadores pueden ser usuarios.

2. Generador de Rotaciones.
   - Generación de las rotaciones diarias atendiendo a los requerimientos según normativa.

3. Gestión de Rotaciones.
   - Guardado de la rotación generada como registro.

4. Gestión de Operarios y Coordinadores.
   - Creación de Operarios y Coordinadores.
   - Actualización de Operarios y Coordinadores.
   - Búsqueda de Operarios y Coordinadores.
   - Eliminación de Operarios y Coordinadores.

5. Listado de Rotaciones.
   - Visualizado de las rotaciones guardadas en una tabla.
   - Búsqueda de rotaciones filtradas por equipo y fecha.

2.2. Requisitos técnicos

Hardware: El hardware incluye todo el equipo necesario para ejecutar un producto, servicio o tecnología. Esto puede incluir computadoras, dispositivos móviles, servidores, redes y otros dispositivos. En este caso en concreto, el hardware necesario para el correcto funcionamiento, seria:

1. Un ordenador accesible por el Coordinador, que disponga de como mínimo de dicho hardware:
   - Una pantalla mínima de 10".
   - Resolución mínima de pantalla de 800x600.
   - Teclado y ratón.
   - Procesador a 1 GHz o más rápido.
   - 2 GB de RAM para 64 bits.
   - Hasta 20 GB de espacio en disco duro disponible.

2. Una red local que facilite el acceso desde la terminal a la Base de Datos del cliente por parte de la aplicación.

3. Un servidor con acceso a la red local.

Software: El software incluye el sistema operativo necesario para ejecutar una aplicación, así como los programas y herramientas necesarias para su funcionamiento. Esto puede incluir lenguajes de programación, bases de datos, servidores web, controladores y software de terceros. En este caso en concreto, el software necesario para el correcto funcionamiento, seria:

- Sistema operativo soportado: Windows 10 64 Bits.
- Instalación de XAMPP 8.0.30 para Windows.
- Apache NetBeans 13 para Windows.
- JDK 15.
- mysql-connector-j-8.2.0.jar
- Base de datos SQL.

Diseño de sistema: El sistema se compondrá de una base de datos relacional con la información y un software que acceda a la misma mediante una red local. El software se diseñara siguiendo el patrón de diseño MVC (Modelo-Vista-Controlador) por ser un patrón que facilita la escalabilidad, posibilita las actualizaciones visuales y permite una mejor organización de las diferentes partes del sistema.

Requisitos de seguridad: Los requisitos de seguridad también son importantes para garantizar la seguridad de los usuarios y los datos. Esto incluye la configuración de seguridad, como el uso de contraseñas. En este caso en concreto, los requisitos de seguridad necesarios para el correcto funcionamiento serian:

- Acceso al terminal con la aplicación solo a personal autorizado.
- Necesario perfil de acceso a la aplicación mediante contraseña.
- Creación de perfil de acceso a la aplicación solo por parte del personal autorizado.

Funcionalidad y rendimiento: Estos son otros requisitos técnicos importantes. Estos incluyen el nivel de funcionalidad y rendimiento que el producto, servicio o tecnología debe tener. Esto puede incluir la velocidad de respuesta, la estabilidad. En este caso en concreto, los requisitos funcionales y de rendimiento necesarios para el correcto funcionamiento serían:

- Escalabilidad de la aplicación.
- Posibilidad de actualizar las interfaces.
- Aplicación liviana.

3. Desarrollo de la aplicación de generación de rotaciones para líneas de montaje

3.1. Herramientas utilizadas

Las principales herramientas utilizadas para la implementación de la aplicación son:

- **Apache NetBeans 13.**

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

- **Base de Datos MySQL**

MySQL es un sistema gestor de bases de datos. Se trata de un sistema de libre distribución y de código abierto. Aclarando lo anterior, se puede descargar libremente y cualquier programador puede mejorar el código de la aplicación.

Otras herramientas:
- Xampp
- Workbench
- Photoshop

Lenguajes:
- Java
- SQL

Patrones de Diseño:
- Patrón de diseño DAO
- Patrón de diseño MVC

Dependencias/Librerías:
- AbsoluteLayout
- JCalendar 1.4
- Mysql-connector 8.2

3.2. Metodología de desarrollo

Teniendo en cuenta el tiempo de desarrollo y el tipo de software a desarrollar que en su totalidad es desarrollado por una sola persona y un cliente tipo del cual ya se saben las exigencias, se ha seleccionado para este proyecto la metodología XP, la cual consta de 4 fases: Planificación, Diseño, Desarrollo, Pruebas.
Esta metodología resulta útil para este proyecto, ya que aumenta la productividad durante su desarrollo.

3.3. Planificación

En esta primera fase, se concretaron los requerimientos funcionales y no funcionales de la aplicación y la gestión de recursos técnicos, físicos y humanos. Además se realizó una previsión de carga de trabajo y la programación de tareas.

3.3.1. Requisitos funcionales

- El sistema debe permitir el inicio de sesión a los usuarios.
- El sistema debe permitir poder cambiar la contraseña al usuario, si facilita los datos de control.
- El sistema debe permitir crear usuarios, siempre que la persona que lo cree sea coordinador.
- El sistema debe indicar que usuario está conectado en todo momento.
- El sistema debe permitir al usuario generar y guardar rotaciones.
- El sistema debe guardar esas rotaciones con los datos del coordinador que las generó.
- El sistema debe permitir al usuario agregar, eliminar, consultar y modificar perfiles de trabajadores para tener un mejor control del personal.
- El sistema debe permitir en todos los casos donde se ingresen datos que se puedan limpiar los mismos.
- El sistema debe permitir al usuario moverse por las diferentes interfaces.
- El sistema debe permitir al usuario buscar rotaciones filtrándolas por fecha y equipo.
- El sistema debe permitir al usuario visualizar las búsquedas de rotaciones en una tabla.
- El sistema debe avisar al usuario si ha cometido algún error o si la acción se ha realizado con éxito.
- El sistema debe permitir al usuario desconectarse de la aplicación en todo momento.
- El sistema debe permitir al usuario salir de la aplicación en todo momento.

3.3.2. Requisitos no funcionales

- Garantizar la confiabilidad, seguridad y el rendimiento de la aplicación.

3.3.3. Recursos físicos

- Ordenador con todos los periféricos.
- Ordenador que cumpla los requerimientos mínimos para la utilización de las herramientas informáticas.
- Conexión a Internet.

3.3.4. Recursos humanos

Nombre	Héctor
Rol	Analista, programador y diseñador
Responsabilidades	Toma de requerimientos, diseño y programación de la aplicación.

3.3.5. Recursos técnicos

- Licencias de las diferentes herramientas, aunque en nuestro caso concreto todas las herramientas tienen licencias de software libre GNU o BSD o son de libre distribución.

3.3.6. Previsión de carga de trabajo y programación de tareas

Concretados ya todos los requisitos y recursos necesarios, podemos realizar una programación y distribución de tareas que nos facilitará poder tomar decisiones informadas sobre el proyecto. Para ayudarnos con dicha tarea utilizaremos el diagrama de Gantt. Toda esta información ayudará a identificar posibles problemas, ajustar el cronograma y asignar recursos de manera más eficiente.
Distribuiremos todas las tareas en tres fases: la fase 1 realizara tareas previas al desarrollo, la fase 2 se encargará del desarrollo y la fase 3 de la documentación.

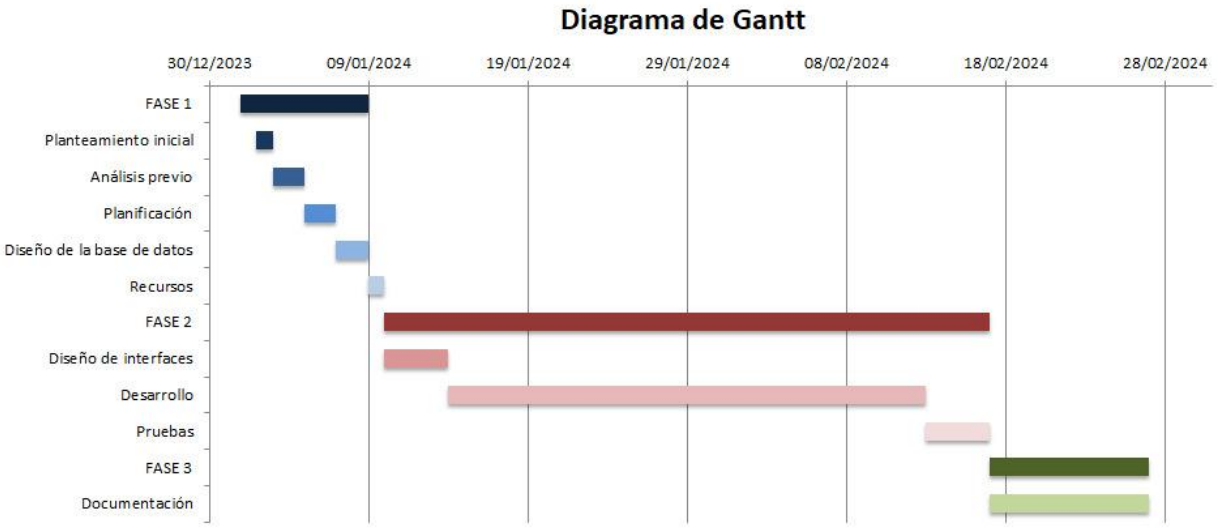
Distribución de horas por tareas:

Tarea	**Descripción**	**Fase**	**Tiempo estimado**
Planteamiento inicial.	Descripción del proyecto y justificación	1	1h
Análisis previo	Solución propuesta y requisitos técnicos	1	2h
Planificación	Concreción de requerimientos, recursos y gestión de los mismos.	1	2h
Diseño de la base de datos	Diseño de la estructura de la BD e implementación de la misma.	1	2h
Recursos	Descarga e instalación de recursos necesarios	1	1h
Diseño de Interfaces	Fijar un patrón de diseño útil y acorde a la estética del cliente	2	4h
Desarrollo	Programación de la aplicación	2	30h
Pruebas	Realización de pruebas sobre las diferentes casuísticas	2	4h
Documentación	Realización del manual técnico, manual de usuario y documentación	3	10h

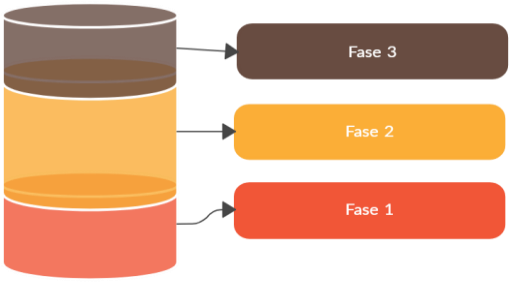
Distribución de horas por fases:

Fase	**Tiempo estimado**
1	8h
2	38h
3	10h
Total	**56h**

Diagramas de Gantt:



Representación de las tareas con el diagrama de Gantt.



Representación del tiempo de desarrollo por volumen.

3.4. Diseño

3.4.1. Casos de uso

Caso de uso: Iniciar sesión

Objetivo	Validar su nombre de usuario y contraseña para iniciar sesión.
Actores	usuario
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1. El actor escribe el nombre de usuario y la contraseña, luego el actor da clic en el botón "Iniciar".	2. Se comprueba si existe el usuario ingresado en la base de datos y valida su respectiva contraseña.
	3. Muestra el diseñador de rotaciones al usuario que inicio la sesión.
FLUJO ALTERNATIVO	
Acción del actor	**Respuesta del sistema**
Punto 1	El actor no digito la información en alguno de los dos campos que se solicita o la escribió erróneamente; el sistema informa mediante una alerta que no existe ese usuario o contraseña y limpia los campos si se digito algo.

Caso de uso: Cambiar contraseña

Objetivo	Permitir al usuario cambia su contraseña
Actores	usuario
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1. El actor hace clic en el botón "Olvidaste la contraseña?"	2. Muestra la página de cambio de contraseña
3. El actor escribe el nombre de usuario, el número de trabajador, la contraseña nueva dos veces, luego el actor da clic en el botón "Cambiar".	4. Se comprueba si existe el usuario y el número de trabajador ingresados en la base de datos y que la contraseña nueva se ha escrito igual las dos veces.
	5. El sistema actualiza la contraseña del usuario e informa que se ha realizado el cambio.
FLUJO ALTERNATIVO	
Acción del actor	**Respuesta del sistema**
Punto 3	El actor no digito la información en alguno de los campos que se solicita o la escribió erróneamente; el sistema informa mediante una alerta que algún dato insertado es erróneo.

Caso de uso: Crear usuario

Objetivo	Permitir a un trabajador (coordinador) crearse un perfil de usuario
Actores	coordinador
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1. El actor hace clic en el botón “Inscribirse”	2. Muestra la página de creación de usuario
3. El actor escribe el nombre de usuario, el número de trabajador, el teléfono y la contraseña dos veces, luego el actor da clic en el botón “Inscribirse”.	4. Se comprueba si existe en la base de datos un coordinador con los datos ingresados y que la contraseña se ha escrito igual las dos veces.
	5. El sistema crea un nuevo usuario.
FLUJO ALTERNATIVO	
Acción del actor	**Respuesta del sistema**
Punto 3	El actor no es un coordinador; el sistema informa mediante una alerta del error.

Caso de uso: Generar rotación

Objetivo	Permitir a un usuario generar una rotación.
Actores	usuario
Precondiciones	Haber iniciado la sesión (Iniciar sesión).
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1. El actor hace clic en el botón “Generar”	2. Muestra la misma interfaz los datos de la rotación generada.

Caso de uso: Crear trabajador

Objetivo	Permitir a un usuario crear un perfil de trabajador
Actores	usuario
Precondiciones	Haber iniciado la sesión (Iniciar sesión).
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1.El actor hace clic en el botón “Trabajador”	2. Muestra la página de CRUD trabajador
3. El actor escribe el nombre del trabajador (tiene que ser una sola palabra), el número del trabajador (tiene que ser un numero), los apellidos (dos palabras separadas por un espacio), el teléfono (nueve dígitos sin separaciones), la categoría (es obligatorio asignar una) y si la categoría es operario los puestos que se sabe (como mínimo 3), luego el actor da clic en el botón “Guardar”.	4. El sistema crea un nuevo trabajador.
FLUJO ALTERNATIVO	
Acción del actor	**Respuesta del sistema**
Punto 3	El trabajador ya existe o algún campo se rellenó mal; el sistema limpia los campos e informa con una alerta.

Caso de uso: Consultar trabajador

Objetivo	Permitir a un usuario consultar información de un perfil de trabajador
Actores	usuario
Precondiciones	Haber iniciado la sesión (Iniciar sesión).
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1.El actor hace clic en el botón “Trabajador”	2. Muestra la página de CRUD trabajador
3.El actor escribe el número del trabajador que quiere buscar, luego el actor da clic en el botón “Buscar”.	4. Muestra toda la información del trabajador.
FLUJO ALTERNATIVO	
Acción del actor	**Respuesta del sistema**
Punto 3	El trabajador no existe; el sistema limpia los campos e informa con una alerta.

Caso de uso: Eliminar trabajador

Objetivo	Permitir a un usuario borrar un perfil de trabajador
Actores	usuario
Precondiciones	Haber iniciado la sesión (Iniciar sesión).
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1. El actor hace clic en el botón “Trabajador”	2. Muestra la página de CRUD trabajador
3. El actor busca un trabajador	4. (Consultar trabajador)
5. El actor da clic en el botón “Borrar”.	6. El sistema elimina al trabajador.

Caso de uso: Actualizar trabajador

Objetivo	Permitir a un usuario actualizar un perfil de trabajador
Actores	usuario
Precondiciones	Haber iniciado la sesión (Iniciar sesión).
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1. El actor hace clic en el botón “Trabajador”	2. Muestra la página de CRUD trabajador
3. El actor busca un trabajador	4. (Consultar trabajador)
5. El actor actualiza cualquiera de los campos(a excepción del número de trabajador) con los condicionantes de ingreso de datos (Crear trabajador), luego da clic en el botón “Actualizar”.	6. El sistema actualiza al trabajador.
FLUJO ALTERNATIVO	
Acción del actor	**Respuesta del sistema**
Punto 3	Los datos insertados son erróneos; el sistema informa con una alerta.

Caso de uso: Consultar rotación

Objetivo	Permitir a un usuario buscar rotaciones
Actores	usuario
Precondiciones	Haber iniciado la sesión (Iniciar sesión).
FLUJO BÁSICO	
Acción del actor	**Respuesta del sistema**
1. El actor hace clic en el botón “Registro”.	2. Muestra la página de registro de rotaciones.
3. El actor selecciona una fecha en el calendario y un equipo, luego da clic en el botón “Buscar”.	4. El sistema los datos de las rotaciones que encajan con los filtros estipulados por el actor.

3.4.2. Diseño de la base de datos

3.4.2.1. Introducción

El propósito general de las bases de datos empleadas en empresas que tienen líneas de producción, es mayoritariamente de carácter administrativo, facilitando el acceso a la información a empleados de dicho departamento a realizar tareas tales como: nominas, seguimiento de horas, etc. En el caso concreto que nos ocupa, la base de datos que se ha diseñado pretende ser un reflejo de estos casos generales, pero obviando muchos datos que no son necesarios para la finalidad de la aplicación.

En el diseño de la base de datos el diagrama Entidad-Relación ha sido de crucial ayuda, al proporcionar una representación visual de las entidades, sus atributos y las relaciones entre ellas, facilitando la comprensión y planificación efectiva de la estructura de la base de datos.

3.4.2.2. Descripción de las entidades

En nuestro diagrama entidad-relación, hemos identificado seis entidades clave que representan las principales categorías de información en nuestra base de datos.

- La entidad '**trabajador**' almacena información esencial sobre los empleados, incluyendo su identificador, número de trabajador, nombre, apellidos, número de teléfono, categoría profesional y el equipo al que pertenece.
- La entidad '**usuario**' registra detalles esenciales sobre los usuarios que van a acceder a la aplicación, incluyendo identificador de trabajador, alias y contraseña.
- La entidad '**operario**' captura información sobre los puestos que saben realizar los trabajadores, como cada uno de los cinco puestos que componen una sección de trabajo.
- Las entidades '**categoría**' y '**equipo**' desglosan aún más los trabajadores, catalogándolos por categoría y agrupándolos por equipos de trabajo.
- La entidad '**rotación**' almacena datos acerca de las rotaciones generadas, como su identificador, número de rotación, identificador del equipo que la ha realizado, fecha, identificador del coordinador que la ha guardado y los cinco números de operario que han trabajado durante esa rotación.

3.4.2.3. Relaciones entre las entidades

Las relaciones entre las seis entidades en nuestro diagrama entidad-relación son fundamentales para comprender la interconexión de la información en nuestra base de datos SQL. Aunque en el mismo se muestren todas las relaciones entre las tablas como uno a muchos, esto es debido a que en SQL no existe como tal la relación uno a uno, por lo que para conseguir emularlo se han aplicado restricciones y reglas tanto en SQL como en la propia aplicación java.

La entidad '**trabajador**' está vinculada a la entidad '**usuario**' mediante una relación uno a uno, ya que un trabajador puede ser un usuario. Asimismo, la entidad '**operario**' está conectada a '**trabajador**' en una relación uno a uno, indicando que un trabajador solo puede tener un perfil de operario y un operario obligatoriamente tiene que un ser trabajador. Además, la entidad '**trabajador**' tiene relaciones uno a muchos con las entidades '**categoría**' y '**equipo**', lo que permite categorizar a los trabajadores y asociarlos a un equipo específicos.

La entidad '**rotación**' también tiene una relación con '**trabajador**' y '**equipo**', indicando qué coordinador guardo la rotación y sobre que equipo se hizo la misma. Estas relaciones bien definidas proporcionan una estructura coherente para el flujo de información en la base de datos, asegurando una gestión eficiente y precisa de los datos.

En nuestra estructura de base de datos, hemos definido claves primarias y foráneas para establecer relaciones sólidas entre las entidades. La entidad '**trabajador**' utiliza un identificador único llamado 'id_trabajador' como su clave primaria, mientras que la entidad '**operario**' no cuenta con su propia clave primaria. La relación entre estas dos entidades se establece mediante la clave foránea 'id_operario' en la entidad '**operario**', la cual hace referencia al 'id_trabajador' en la entidad '**trabajador**', creando así una conexión directa entre trabajadores y sus funciones si son operarios.

La entidad '**usuario**' funciona similar a la relación '**operario**'-'**trabajador**'. La relación se establece con la clave foránea 'id_trabajador' en la entidad '**usuario**', la cual hace referencia al 'id_trabajador' en la entidad '**trabajador**'.

En el caso de las entidades '**categoría**' y '**equipo**', ambas cuentan con clave primaria: 'id_categoria' y 'id_equipo'. La entidad '**trabajador**' utiliza las claves foráneas 'categoria' y 'equipo' para relacionarse con la categoría profesional y los equipos de trabajo.

Por último, la entidad '**rotación**' establece una relación con '**trabajador**' y '**equipo**' mediante las claves foráneas 'coordinador' y 'id_equipo', que referencia a las claves primarias 'id_trabajador' en la entidad '**trabajador**' y 'id_equipo' en la entidad '**equipo**'. Esta relación rastrea qué coordinador ha generado la rotación y sobre qué equipo de trabajo, proporcionando información valiosa sobre la gestión de operarios y la responsabilidad del personal en el sistema.

3.4.2.4. Restricciones y reglas de negocio

En nuestro diseño de base de datos SQL, hemos incorporado diversas restricciones y reglas de negocio para garantizar la integridad y coherencia de los datos.

- La entidad '**trabajador**' cuenta con varias restricciones que aseguran la unicidad del número de trabajador y la obligatoriedad de tener un ID de trabajador, nombre, apellidos y teléfono, además, se estable como categoría por defecto la de operario y al identificador como clave auto incremental de la entidad, evitando con todo ello la duplicación de información clave. También posee una restricción de integridad referencial que garantiza que todos los trabajadores pertenecen a una categoría y a un equipo, aunque en este último caso puede tener valor nulo.
- La entidad '**usuario**' posee varias reglas que aseguran la unicidad del ID del trabajador asociado y la obligatoriedad de tener tanto el identificador como un alias y una contraseña. Además, de sumar una restricción de integridad referencial que garantiza que todos los usuarios están asociados a un trabajador.
- La entidad '**operario**' tiene una restricción de integridad referencial que impide que puedan existir operarios que no sean trabajadores y una restricción aseguran la unicidad del ID del trabajador haciendo que un trabajador solo pueda tener un perfil de operario, asegurando la consistencia en la gestión del personal.
- La entidad '**categoría**' cuenta con varias restricciones que aseguran la unicidad del nombre de la categoría y la obligatoriedad de tener un ID de categoría como clave de entidad y un nombre, evitando posibles conflictos por duplicidad.
- La entidad '**equipo**' posee las mismas restricciones que 'categoría', asegurando la unicidad del nombre de equipo y la obligatoriedad de tener un ID de equipo como clave de entidad y un nombre, evitando igualmente posibles conflictos por duplicidad.
- La entidad '**rotación**' tiene varias restricciones que aseguran la obligatoriedad de tener un ID de rotación, numero de rotación, ID del equipo que realiza la misma, la fecha, ID del coordinador y los cinco números de trabajadores que cubres los 5 puestos de esa sección, además el ID de la rotación se establece como clave auto incremental de la entidad, evitando como ya se ha comentado antes la duplicación de información clave. También posee una restricción de integridad referencial que garantiza que todas las rotaciones se han realizado por un coordinador y con un equipo determinado.

Estas restricciones y reglas de negocio no solo protegen la calidad de los datos, sino que también cumplen con requisitos específicos de negocio, proporcionando una base sólida para la confiabilidad de la información en nuestra base de datos.

3.4.2.5. Diagrama Entidad-Relación

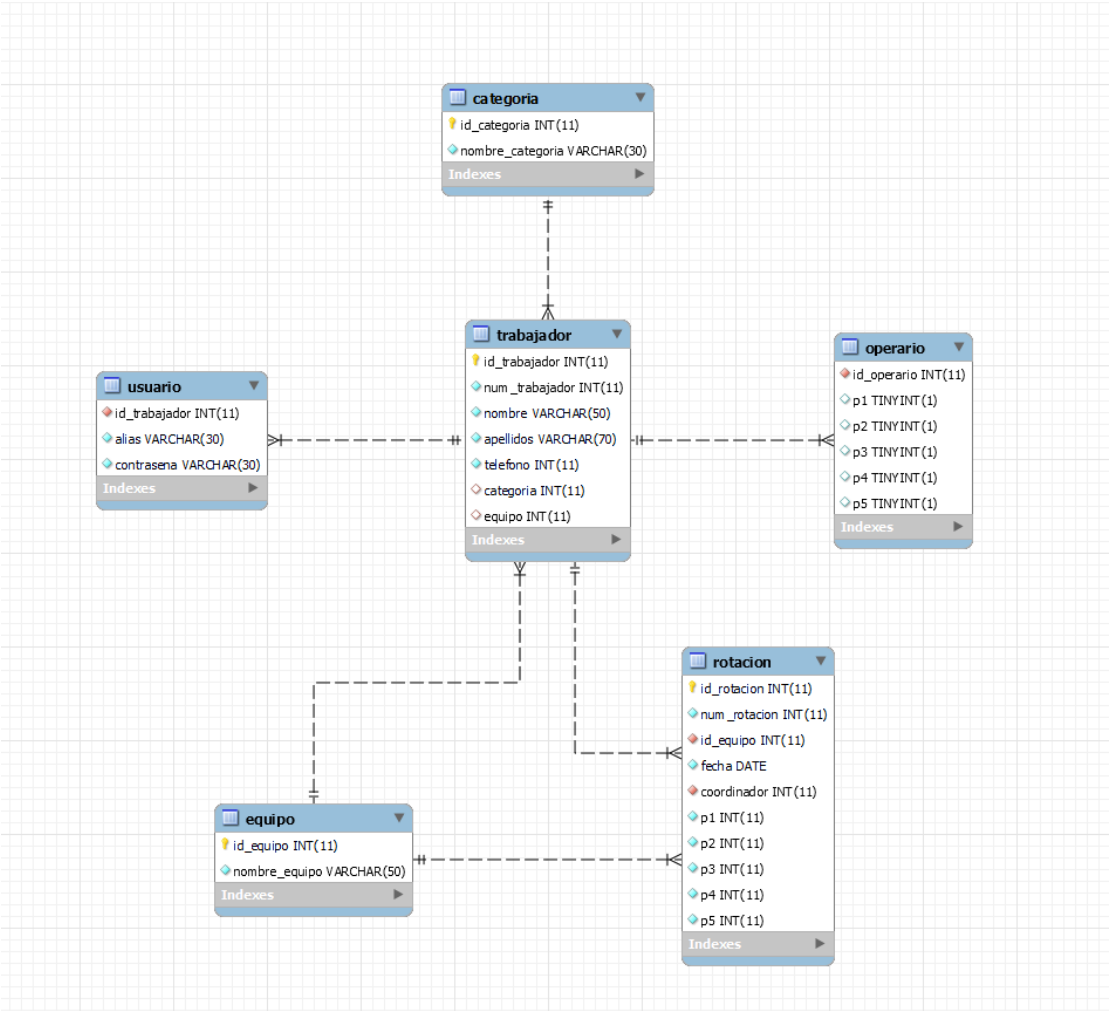


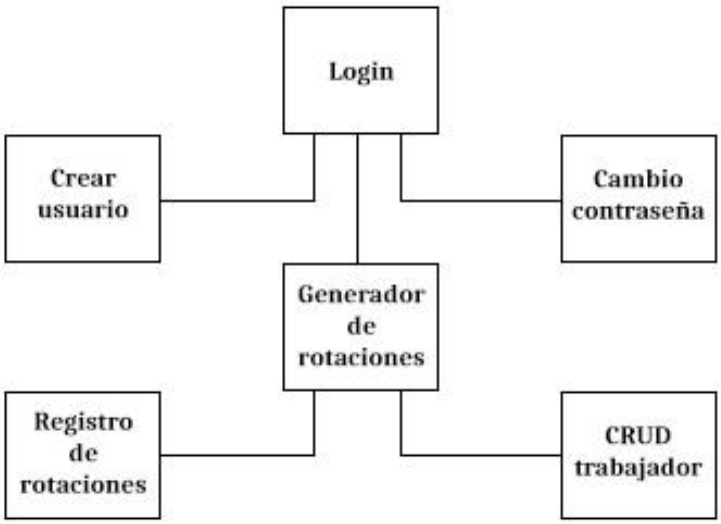
Diagrama Entidad-Relación de la Base de datos creada.

3.4.3. Diseño de interfaces

Se define la manera de cómo es la estructura visual de la aplicación, la justificación técnica de como aparecerán los objetos navegacionales en la interfaz y las tecnologías necesarias empleadas en el diseño de las interfaces.

3.4.3.1 Diseño navegacional

Se presenta un mapa de navegación en el cual se aprecian cada una de las vistas. Como elementos añadidos para facilitar la navegabilidad de la aplicación, se han implementado componentes que nos permiten volver a la vista anterior, esto con el fin de que el usuario pueda desplazarse fácilmente por la aplicación.



Mapa de navegación de la aplicación.

3.4.3.2. Usabilidad

La usabilidad es crucial en el diseño de interfaces, ya que se centra en la experiencia del usuario, facilitando la interacción efectiva y eficiente con un sistema. Una interfaz usable mejora la satisfacción del usuario, reduce la curva de aprendizaje y promueve la fidelidad, influyendo directamente en el éxito y la adopción de una aplicación o sitio web.

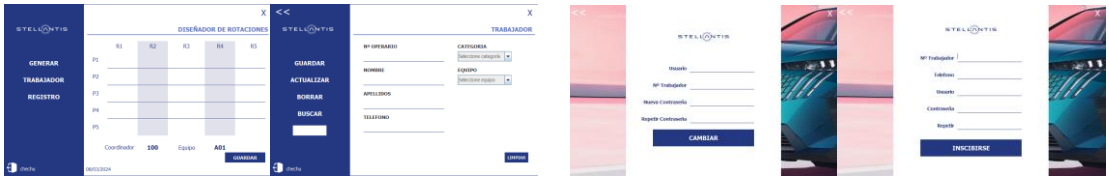
Se han diseñado las interfaces atendiendo a los principios de usabilidad:

- **Simplicidad:** Se ha presentado la información y las funciones de manera clara y sin complicaciones innecesarias. Con un diseño simple facilitando la comprensión y el uso, mejorando la eficiencia y la satisfacción del usuario.

Ejemplo: Elementos interactivos con colores contrastados.

- **Consistencia:** Se han seguido patrones de diseño de manera uniforme en toda la interfaz. Manteniendo una apariencia y comportamiento coherentes para facilitar la predicción del usuario y aumentando la usabilidad al proporcionar una experiencia más intuitiva y predecible. Uno de los factores que más influye en la consistencia, es la gama cromática y las hojas de estilo. Normalmente las empresas suelen tener ya esa documentación e incluso la facilitan de manera gratuita.

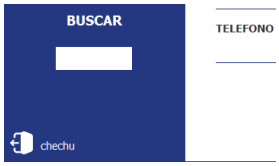
Ejemplo: Patrón de color de la empresa cliente.



Ejemplo: Se mantienen los mismos patrones de diseño para zonas con similar temática.

- **Retroalimentación:** Los usuarios son informados sobre el resultado de sus acciones. Se logra mediante elementos visuales que indican si una acción ha sido completada con éxito o si se requiere alguna corrección. La retroalimentación inmediata refuerza la comprensión del usuario.

Ejemplo: Diferencias visuales para informar si un elemento esta pulsado.

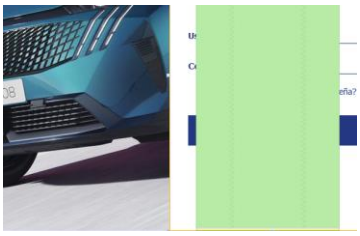


Ejemplo: Visualización del usuario conectado.

3.4.3.3. Tecnologías

Las tecnologías empleadas para el diseño de las interfaces han sido como ya se ha comentado previamente, el lenguaje de programación Java utilizando NetBeans como IDE. Además, para facilitar la disposición de los diferentes elementos, se ha empleado un diseño absoluto.

AbsoluteLayout permite colocar componentes en posiciones arbitrarias en el área de diseño. Las posiciones se especifican en el método *addComponent*() con coordenadas horizontales y verticales relativas a un borde del área de diseño. Esta flexibilidad nos facilita colocar componentes en posiciones que de otra manera no podríamos con el diseño estándar.



Ejemplo: Colocación de componentes a sangre.

También se ha seleccionado JCalendar como selector de fechas proporcionándonos varios componentes visuales que mejoran las experiencia del usuario, y algunas clases que nos ayudan a acceder al mismo.

JCalendar

3.4.3.4. Adaptabilidad y responsividad

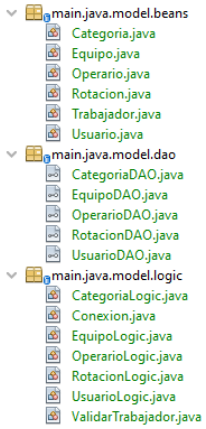
La decisión de no hacer una aplicación de software adaptativa o responsiva está basada en el contexto específico en el que se ha desarrollado.
La aplicación está destinada exclusivamente a ser utilizada en unos dispositivos o contextos específicos y no se espera su uso en diferentes plataformas o tamaños de pantalla.

3.5. Desarrollo

Durante la fase de desarrollo, se implementó el patrón Modelo-Vista-Controlador (MVC) para estructurar y organizar la aplicación de manera eficiente. En este enfoque, el código se divide en tres grupos principales organizados en paquetes:

3.5.1. Modelo

Se utilizó Java como lenguaje principal para la lógica del modelo, encargado de gestionar los datos y las interacciones con la base de datos. Se crearon clases para representar las entidades de la base de datos con relaciones de herencia para emular sus relaciones y facilitar así tanto la comprensión, como el manejo de los datos. La implementación del patrón DAO (Data Access Object) se integró en el modelo para acceder a los datos de manera organizada y eficiente desde la base de datos.



Distribución de los elementos del modelo.

3.5.2. Vista

Java Swing fue empleado para definir la estructura y presentación de las vistas de la aplicación. Los elementos como imágenes o logotipos se han agrupado para un fácil acceso por parte de las vistas. La separación clara entre el modelo y la vista permitió una gestión más efectiva de la interfaz de usuario, facilitando futuras actualizaciones y modificaciones en el diseño.

Vistas y Recursos empleados para las mismas.

3.5.3. Controlador

Los controladores implementados, se encargan de gestionar las interacciones del usuario con los diferentes elementos de la aplicación y actuar como intermediario entre el modelo y la vista. Se han utilizado diferentes funciones para mejorar la experiencia del usuario y gestionar los diferentes 'inserts' para un correcto funcionamiento.

Antes de la codificación, se realizó un diseño previo que clarificó los requerimientos de la aplicación. Este enfoque pro-activo permitió una implementación más efectiva y una mejor comprensión de la estructura general del sistema. La utilización del patrón MVC no solo organizó el código de manera modular, sino que también facilitó el mantenimiento y la escalabilidad de la aplicación a medida que evolucionaba con nuevos requisitos y funcionalidades.

3.5.4. Algoritmo del generador de rotaciones

El algoritmo de generación de rotaciones se implementó para recolectar y mostrar los números de trabajadores de un equipo, que ocuparan cada uno de los puestos durante las cinco rotaciones que harán durante su jornada diaria.
Dividiremos el algoritmo en 2 partes diferenciadas:

- La primera se encarga de buscar los cinco trabajadores que ocupen los cinco puestos durante una jornada. Funciona buscando en primera instancia un trabajador aleatorio que pueda realizar el primer puesto, luego a un trabajador aleatorio que pueda realizar el segundo puesto, pero omitiendo al trabajador seleccionado anteriormente. Y continuaremos esta dinámica hasta completar los cinco puestos. Si el sistema detectara que no hay trabajadores que puedan cubrir un puesto volverá a realizar las búsquedas nuevamente. Realizara esta acción para cada una de las 5 rotaciones.

- La segunda se ocupara de comparar cada una de las rotaciones contiguas para asegurar que un trabajador no repita puesto durante dos rotaciones seguidas. Si el sistema detecta que se ha dado el caso, repetirá la primera parte hasta que se cumpla la condición.

4. Pruebas

Opté por no limitar las pruebas exclusivamente al final del proceso, sino que las fui llevando a cabo de manera incremental a medida que desarrollaba los diversos métodos a través de pruebas unitarias. Esta elección tuvo como finalidad reducir la cantidad de errores que debería abordar al término del proceso y la propagación de los mismos, facilitando el proceso de desarrollo y corrección.
Además, en la fase final del desarrollo se realizaron pruebas funcionales para asegurar que todas las características de la aplicación funcionaban correctamente. Por otro lado, se controlaron los posibles errores de acceso a la base de datos.

5. Conclusiones

Cumplimiento de la Planificación:
Tras evaluar el desarrollo del proyecto, es evidente que se ha logrado un alto grado de cumplimiento de la planificación establecida. La mayoría de las tareas se llevaron a cabo según lo programado, y los hitos importantes se alcanzaron siguiendo con los tiempos de producción.

Aprendizajes Adquiridos:
Durante este proyecto, se ha adquirido un conocimiento significativo en áreas clave como la programación en Java, diseño de bases de datos y gestión de proyectos de software. La necesidad de implementar determinados recursos ha propiciado conocer librerías nuevas, sus componentes y métodos para interactuar con ellos. En definitiva una experiencia muy enriquecedora que más haya de adquirir conocimientos para solventar problemas, ha servido como catalizador para aprender más y ser más eficientes.

Desafíos Enfrentados:
Uno de los mayores desafíos encontrados fue la implementación del algoritmo. La complejidad inherente resultó ser un aprendizaje valioso, pero también señaló la necesidad de una planificación más detallada y la asignación de recursos específicos para superar obstáculos técnicos.

Posibles Mejoras:
Después del desarrollo, con la experiencia adquirida y más tiempo, se podría cambiar el algoritmo de generación de rotaciones por uno más eficiente o más flexible. Además, se podría implementar futuros módulos como la creación de informes, el envío de los mismos al correo de cargos superiores o a personal o creación de estadísticas y visualización de las mismas.

Refinamiento de la Planificación:
Identificar y abordar las áreas de la planificación que demostraron ser más flexibles o menos precisas. Una mayor atención a la estimación de tiempos y riesgos podría mejorar futuros procesos de desarrollo.

Perspectivas Futuras:
Basándonos en las lecciones aprendidas y las mejoras propuestas, se puede valorar el comercializar e implementar esta herramienta en otras empresas de la automoción o de cualquier empresa que cuente con líneas de producción. Además, la experiencia acumulada servirá como una base sólida para enfrentar desafíos técnicos y gestionar de manera más eficiente los recursos y el tiempo en proyectos venideros.

6. Presupuesto

Desglose de Tareas:

Tarea	**Tiempo estimado**
Planteamiento inicial.	1h
Análisis previo	2h
Planificación	2h
Diseño de la base de datos	2h
Recursos	1h
Diseño de Interfaces	4h
Desarrollo	30h
Pruebas	4h
Documentación	10h
Total	56h

Recursos Humanos:

El coste de los recursos humanos de este proyecto, se limitara a incluir los costes por hora del desarrollador que ha realizado dicha aplicación.

Función	**Coste (euros/hora)**	**Horas**	**Total**
Analista-programador-diseñador	40	56	2.240 euros

Herramientas y Tecnologías:

Al utilizarse herramientas de desarrollo de software libre y tener el equipo de hardware amortizado, solo se aplicara una cuantía en forma de porcentaje añadido al total del coste del proyecto, como forma de sufragar el deterioro del material por uso. Este porcentaje se establece en un 3%.

Contingencias:

Se han añadido una reserva para imprevistos o cambios en los requisitos del proyecto. El porcentaje que se ha estipulado es un 10%, e incluiría cambios que no supusieran un aumento del presupuesto en más del 5%.

Total del Presupuesto:

Recursos Humanos	**Herramientas y Tecnologías**	**Contingencias**	**Total**
2.240 euros	67,2 euros	224 euros	2.531,2 euros
I.V.A. (21%)			531,56 euros
Precio final			**3.062,75 euros**

7. Bibliografía

API de Java Java
https://docs.oracle.com/javase/8/docs/api/

Eventos del Mouse en Java – MouseEvent Fredy Geek
https://fredygeek.com/2020/09/06/eventos-del-mouse-en-java-mouseevent/

Brand Color Codes Stellantis
https://www.brandcolorcode.com/stellantis

JCalendar Java date chooser bean
https://toedter.com/jcalendar/

Collentions.shuffle
https://www.geeksforgeeks.org/collections-shuffle-method-in-java-with-examples/

El patrón modelo-vista-controlador (MVC) Universidad Complutense de Madrid
https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.mvc.pdf

Data Access Object (DAO) Oscar Blancarte
https://reactiveprogramming.io/blog/es/patrones-arquitectonicos/dao

DAO (DATA ACCESS OBJECT)
https://miprimerblog-temas.blogspot.com/2013_07_01_archive.html

AbsoluteLayout davidbritch
https://learn.microsoft.com/es-es/dotnet/maui/user-interface/layouts/absolutelayout?view=net-maui-8.0