

Projecte Final de Cicle

Aplicació per a l'organització i seguiment àgil de tasques per a projectes I+D

Alumne:	Alejandro Rubio Rico
DNI:	48646684-M
Tutor Individual:	Alicia Gonzalez Canet
Tutor del grup:	Sebastian Ciscar Breitzler

Datos del proyecto

Datos del alumno

Nombre y apellidos	Alejandro Rubio Rico
NIF/NIE	48646684M
Curso y CF	2o DAM

Datos del proyecto

Título del proyecto	Aplicació per a l'organització i seguiment àgil de tasques per a projectes I+D
Nombre del tutor individual	Alicia González Canet
Nombre del tutor del grupo	Sebastian Ciscar Breitzler
Resumen	<i>El proyecto consta de una aplicación multiplataforma desarrollada en flutter con backend programado mediante springboot, que tiene como objetivo mejorar el actual sistema de planificación y seguimiento de tareas diarias y semanales bajo premisas de un sistema de organización ágil en un entorno de desarrollo de proyectos I+D en el ámbito de la energía. Tecnologías utilizadas: Flutter, MySQL/PostgreSQL, SpringBoot, Java, Dart.</i>
Abstract	<i>The project consists of a multiplatform application developed in flutter with backend programmed using Springboot, which aims to improve the current system of planning and monitoring of daily and weekly tasks under the premises of a system of agile organization in an environment of development of R & D projects in the field of energy. Technologies used: Flutter, MySQL/PostgreSQL, SpringBoot, Java, Dart.</i>
Módulos implicados	Bases de dades, Desenvolupament d'interfícies, Programació Multimèdia i Dispositius Mòbils, Accés a dades
Fecha de presentación	5 de junio de 2024

índice

Datos del proyecto	1
Datos del alumno	2
Datos del proyecto	2
índice	3
1. Introducción al proyecto	4
1.1. Descripción del proyecto	4
1.2. Objetivos	5
1.3. Tipo de proyecto	6
2. Análisis del sistema actual	7
2.1. Descripción del sistema actual	7
2.2. Viabilidad del sistema actual.....	14
2.3. Requerimientos del nuevo sistema.....	15
3. Análisis de la solución.....	17
3.1. Análisis de las posibles soluciones	16
3.2. Evaluación de las posibles soluciones	18
3.3. Descripción de la solución escogida	19
4. Diseño de la solución	20
4.1. Requisitos.....	20
4.2. Análisis de riesgos.....	23
4.3. Casos de uso.....	24
4.4. Diagramas	25
5. Desarrollo de la solució	30
5.1. Estimación de coste temporal	30
5.2. Dotación de recursos.....	31
5.3. Desarrollo del sistema	32
5.4. Evaluación del sistema	59
6. Implantación de la solución	68
6.1. Fases de implantación.....	68
6.2. Formación, comunicación y soporte	69
6.3. Mejoras y futuras implementaciones.....	70
7. Conclusiones	71
8. Bibliografía	72

1. Introducción al proyecto

1.1. Descripción del proyecto

El proyecto busca diseñar, desarrollar y, de manera limitada y a nivel de trabajo en local, desplegar una aplicación para la gestión de tareas diarias y semanales de un área de trabajadores en una empresa especializada en proyectos I+D en el ámbito de la energía. Para ello, el proyecto parte de la situación actual en la empresa, identificando y caracterizando la necesidad organizativa y de uso de la aplicación, con el fin de traducir esa necesidad en unos requisitos de usuario y, por ende, de funcionalidades que debe presentar dicha aplicación para la función para la que es concebida.

La empresa organiza su trabajo por proyectos I+D mediante una estructura matricial de recursos técnicos que se caracteriza por la polivalencia de éstos, existiendo personas con grados diferentes de responsabilidad dependiendo del proyecto. Además, la variedad e incertidumbre de las tareas a realizar requiere enfocar la planificación y el seguimiento del trabajo mediante un enfoque bajo el paraguas, hasta cierto punto, de metodologías ágiles que permitan una respuesta rápida y práctica a la resolución de problemas combinada con un seguimiento incremental de los objetivos de cada proyecto.

Se busca una herramienta multiplataforma para maximizar las facilidades de acceso a cada una de las personas del área en que se pretende implantar en un inicio, inicialmente pensando en terminales móviles y ordenadores de sobremesa o portátil de la empresa con el fin de facilitar su acceso tanto en el centro de trabajo como fuera de él o durante la realización de trabajos en campo. La aplicación se diseñará teniendo en cuenta los diferentes perfiles de usuario que harán uso de ella, de manera que sirva tanto para la organización del trabajo personal de cada uno como para el seguimiento y detección de desviaciones de los objetivos del proyecto. Se deja fuera de este alcance el seguimiento de alto nivel del proyecto (A nivel de hitos, facturación, sumatorios totales de horas, etc), entre otras cosas porque están cubiertos total o parcialmente por otras aplicaciones.

1.2. Objetivos

Se pretende conseguir los siguientes objetivos:

- Identificar y definir correctamente las necesidades actuales en la empresa
- Describir unos requisitos de la aplicación acorde a dichas necesidades
- Desarrollar una serie de funcionalidades que cubran las citadas necesidades en base a los requisitos
- Diseñar la aplicación a nivel tanto funcional como estético (Interfaces)
- Definir una arquitectura de la aplicación y herramientas (Lenguajes, frameworks) a emplear
- Diseñar y desarrollar la base de datos necesaria
- Desarrollar el sistema backend de gestión de los datos y funcionamiento interno de la lógica de la aplicación y el acceso a los datos de la base de datos, incluyendo la implementación de operaciones CRUD
- Desarrollar las interfaces previamente diseñadas, la navegación, etc hasta el punto en que las limitaciones de tiempo y recursos lo permitan
- Integrar ambas partes de la aplicación y verificar, de manera limitada en cuanto a tiempo de testeo y pruebas, dicha integración

1.3. Tipo de proyecto

El proyecto es de tipo interno, ya que se lleva a cabo por la propia empresa para su aprovechamiento directo y sin ningún tipo de explotación prevista, y es un proyecto de programación (Desarrollo de software). Dicho desarrollo abarca tanto la parte de cliente (Frontend) como al de servidor (Backend). La aplicación final a desarrollar se genera de cero, como alternativa a un método organizativo compuesto por varias aplicaciones que cubren solo parcialmente los requisitos y necesidades y que, en todo caso, se consideran insuficientes para la realización óptima del trabajo que se prevé, tal y como se detallará más adelante.

2. Análisis del sistema actual

2.1. Descripción del sistema actual

En este punto se realiza una revisión del sistema actual con el fin de contextualizar la necesidad de una aplicación para la mejora organizativa, y la estructura que ésta debe seguir en base al método organizativo actual.

Lo primero conviene describir la actividad de la empresa y el sistema actual de organización. La actividad principal de la empresa se da en la ejecución de proyectos I+D en el ámbito de la digitalización y la energía. Para ello, la empresa trabaja en un enfoque a proyectos de períodos variantes entre el corto plazo (semanas o meses) y el largo plazo (Hasta 5 años). El área para la que se diseña la aplicación tendrá previsiblemente en el momento del despliegue 8 empleados, que se pueden clasificar en los siguientes rangos:

- Responsable de área: 1 persona
- Coordinadores de proyecto: 2 personas
- Recursos técnicos: 5 personas

La atribución de tareas y responsabilidades se realiza acorde al siguiente diagrama:



Como se aprecia, a efectos prácticos, el coordinador del área es la persona con mayores atribuciones seguida de los coordinadores de proyecto. Los recursos técnicos se limitan a ejecutar las tareas de proyecto. No obstante, los coordinadores de proyecto también pueden ejecutar tareas en los proyectos, ya sea en los suyos propios o en los de otro coordinador, al

igual que el responsable de área. Sin embargo, se considerará a efectos prácticos en las atribuciones de usuario y permisos que el responsable de área tiene las mismas atribuciones que los responsables de proyecto, ya que al nivel organizativo para el que se plantea la aplicación no se considera necesaria una diferenciación entre esos dos niveles de usuario.

Por otro lado, se ha de considerar que el enfoque a Proyectos implica diferentes niveles de organización, tal y como se recoge en la siguiente tabla:

Horizonte temporal	Acción organizativa	Recursos involucrados	Herramienta(s) empleada(s) actualmente
Diario	Planificación individual de acciones de cada recurso	Todos	Excel online organizativa
Semanal / Bisemanal	Planificación y seguimiento de objetivos a corto plazo del proyecto	Todos, en especial Coordinador de proyecto	Microsoft planner, metodología SCRUM, Excels de seguimiento de plazos y costes (GANTT)
Mensual	Planificación estratégica y reporte del proyecto a mandos superiores	Coordinador de proyecto y responsable de área	Excel general de seguimiento de costes y plazos, consultas a ERP y visualización cuadros de mando BI

Es importante recalcar que la aplicación a desarrollar en este proyecto busca suplir enteramente la aplicación actualmente empleada para la organización diaria (que no deja de ser una hoja Excel online), y complementar el uso de aplicaciones de seguimiento semanal y bisemanal, ya que esta aplicación pretende proporcionar seguimiento del cumplimiento de tareas y tangibles asociados (como se explicará más adelante).

A continuación, se adjunta una captura de pantalla de la Excel organizativa a modo de ejemplo, donde por cuestiones de confidencialidad se difuminan las tareas específicas de los proyectos.

Para más claridad, se expone a continuación el detalle de una casilla diaria de tareas de un trabajador concreto vacía:

Los campos son los siguientes:

- **Capacidad diaria:** horas diarias efectivas de trabajo para ese recurso. Puede oscilar entre 7 y 9 (Según la modalidad de horario que siga el trabajador y el día de la semana). En ocasiones puede ser incluso menor, si el trabajador hace menos horas de las que teóricamente debería, sea por razones personales, sea por motivos justificados (Como puede ser, por ejemplo, una visita al médico).
- **Tarea:** En la columna de la izquierda se irá registrando una descripción breve cada tarea.
- **Horas previstas:** horas que la persona estima que dedicará a la tarea ese día. En esta Excel, para tareas no previstas al inicio del día, el trabajador debe registrar un 0 en esta casilla, que la Excel automáticamente resaltará en rojo.
- **Horas imputadas:** horas que la persona ha dedicado finalmente a la tarea ese día. Para tareas previstas, pero no ejecutadas, la persona deberá ingresar en este campo un 0.
- **GAP o desviación:** es una métrica que realiza la diferencia (con signo) entre planificación y dedicación efectiva. Proporciona información de si la tarea ha sido sobredimensionada, o si se ha realizado en más tiempo del previsto, en caso de existir dicha desviación.
- Filas de **subtotales:** realizan una suma de cada una de las columnas con los siguientes objetivos:
 - **Horas previstas:** en esta casilla se suma el total de horas indicadas en la columna para obtener el total de horas previstas de trabajo para ese día
 - **Horas imputadas:** en esta casilla se suma el total de horas que se ha indicado que se imputa en la columna para obtener el total de horas que se han realizado de trabajo efectivo ese día. Este total debería coincidir con el de horas previstas, salvo excepciones muy contadas.
 - **Capacidad vs plan:** resta el total de horas previstas a la capacidad diaria indicada. Sirve como indicador de ajuste de la planificación.
- **Tipo:** en esta columna se activa un desplegable de proyectos o actividades generales en las que encuadrar cada tarea, tal y como se muestra a continuación (Los proyectos se han censurado por cuestiones de confidencialidad):



Por otro lado, la herramienta previamente mencionada de software que se emplea a nivel organizativo semanal/bisemanal es el Microsoft Planner. Para el área para la que se diseña la herramienta de la empresa, esta herramienta se emplea fundamentalmente con un enfoque de planificación y revisión de sprints de trabajo, tomando el concepto de “Sprint” de la metodología SCRUM, si bien no se aplica más que una adaptación simplificada de dicha metodología:

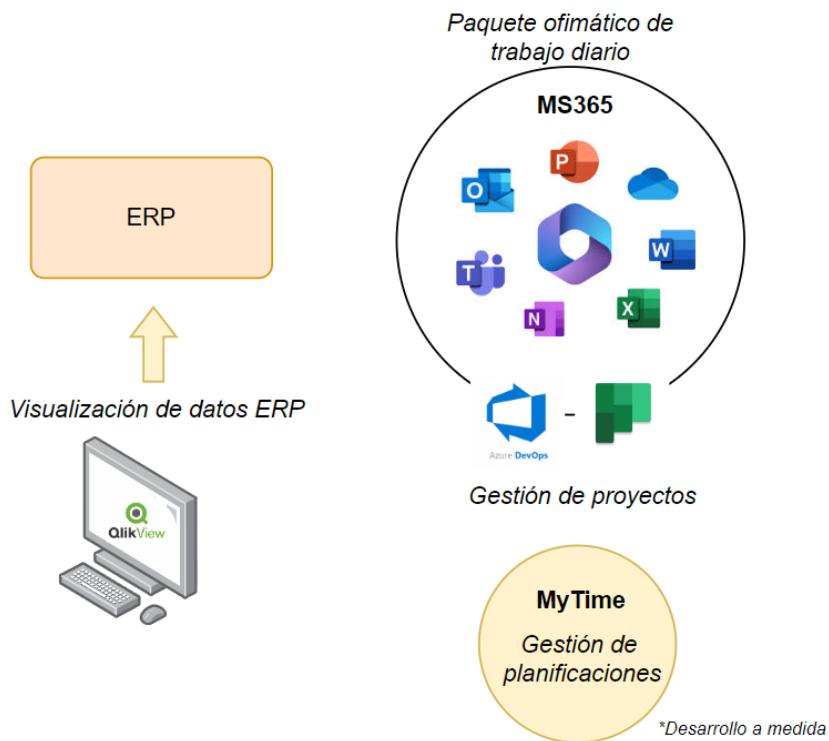
- Sin los roles de SCRUM como tal, siendo únicamente el coordinador de cada proyecto quien ejerce de “Product owner” y de “Scrum master”. El coordinador de proyecto dinamiza y orienta cada reunión de revisión de sprint (Y posterior reunión de planificación del sprint siguiente) para, junto al resto de recursos (Trabajadores técnicos participantes en el proyecto) definir las tareas que pasan a formar parte de la pila de sprint o nuevas tareas a considerar en la pila de producto.
- Empleando pilas de producto únicas o desagregadas, según la necesidad
- Empleando pilas de sprint para cada semana o cada dos semanas, según el proyecto
- Indicando mayor o menor grado de detalle, poniendo o no etiquetas, designando uno o varios responsables para cada tarea, según el caso

A continuación, se muestra una captura de pantalla (Parcialmente censurada para nombres de empresa, conceptos del proyecto y personal de la empresa, por cuestiones de confidencialidad) de la citada herramienta para un proyecto (Plan) concreto:

Backlog - Importante	Backlog - URGENTE	SPRINT - 15/03/24
<input type="button" value="Agregar tarea"/> <input type="radio"/> Informe de medidas	<input type="button" value="Agregar tarea"/> <input type="radio"/> Definir claramente la operatividad y representatividad del equipo de medida de caudal	<input type="button" value="Agregar tarea"/> <input type="radio"/> Presentación GD del área
<input type="radio"/> Comprobación de que el balance mísico y térmico es correcto integrando generación y consumo	<input type="radio"/> Se comprueba en	<input type="radio"/> Vencimiento FB L AA
<input type="button" value="Vencimiento"/> FB JT L <input type="radio"/> Formación interna y sinergias	<input type="button" value="Vencimiento"/> P AA <input type="radio"/> Participación en	<input type="button" value="Vencimiento"/> L <input type="radio"/> Adaptar código tratamiento datos 1732 para potencia
<input type="radio"/> Definición de energéticas y de definición de	<input type="radio"/> Vídeo promocional	<input type="button" value="Vencimiento"/> AR AA <input type="radio"/> Generar código tratamiento datos 1735
	<input type="radio"/> 1 / 3	<input type="radio"/> Vencimiento L <input type="radio"/> Obtener feedback de con
	<input type="button" value="Vencimiento"/> AD	<input type="radio"/> Vencimiento L

Como se ha justificado antes, no se pretende sustituir con la herramienta a desarrollar en este proyecto el uso del Planner, que además está implantado de manera más o menos extendida a toda la empresa, sino que el diseño de ésta se orientará a un uso complementario del Planner, tal y como se describirá más adelante.

A continuación, se muestran un par de diagramas de las diferentes herramientas software que se emplean para la gestión de la actividad operativa de la empresa junto a la jerarquía a nivel de gestión de datos en que se ubican:



- 1 **ERP: Datos de negocio**
- 2 **Gestión de planificaciones** MyTime
- 3 **Gestión de proyectos** 
- 4 **Gestión de la actividad diaria por recurso** Excel, reuniones de seguimiento..

2.2. Viabilidad del sistema actual

El actual sistema tiene una serie de limitaciones que se detallan en la siguiente tabla:

- Cada nuevo mes se debe generar una nueva hoja Excel manualmente adaptando el número de días y fechas concretas, y eliminando los fines de semana
- Cada persona tiene acceso a toda la información del resto sin ninguna limitación
- No existen restricciones a la hora de indicar horas o tareas
- La detección de tareas no previstas es muy básica, debiendo la Excel detectar las casillas de "Horas previstas" puestas a 0.
- La detección de desviaciones puede ser enmascarada por tareas planificadas, pero no realizadas, que tienen igualmente una desviación o gap positivo. En consecuencia, no se puede distinguir cuando una tarea no se ha realizado y cuando se ha realizado una tarea en menos tiempo del previsto. Se tiene un problema igual en el caso contrario; cada tarea no prevista pero realizada genera una desviación negativa que la hace indistinguible de una tarea que se realice en más tiempo del previsto.
- La explotación de los datos y cálculo de indicadores es ardua y todavía no se ha implementado en la Excel por falta de tiempo y de consenso del equipo sobre qué métricas emplear, sobre todo a la hora de cuantificar tareas previstas, pero no realizadas, tareas no previstas, porcentaje de tareas por proyecto, etc...
- Solo se indican tareas de cada proyecto, pero no se indican de qué manera generan incremento en el valor del proyecto. En otras palabras; las tareas se presentan de una manera aislada, sin contextualizar su importancia en el proyecto y como contribuyen a mejorar los objetivos marcados por éste.

Por otro lado, la propia herramienta es la hoja Excel que hace de base de datos, de manera que ante cualquier problema con ésta (Borrado accidental, baja de la empresa de la persona que está alojando dicha hoja Excel en su OneDrive, corrupción del archivo...) se perdería la herramienta y todo el consecuente registro de tareas. Es uno de los principales aspectos a mejorar la persistencia de esos datos.

2.3. Requerimientos del nuevo sistema

Los requisitos que debe cumplir el sistema son los siguientes:

- Debe ser **multiplataforma**, para facilitar el acceso desde el ordenador o cualquier terminal móvil, dado que en ocasiones el personal técnico se encuentra desplazado fuera del centro de trabajo y no tiene acceso al portátil (Puestas en marcha, visitas a planta...), de manera que el hecho de que se pueda acceder también desde el móvil es una ventaja
- Debe desarrollarse una arquitectura que en el futuro permita un **acceso autenticado** con diferentes niveles de usuario y proporcionando funcionalidades específicas según el nivel de perfil de acceso. Se consideran 3 niveles: coordinador de área, coordinador de proyecto y personal técnico general.
- Los datos se **almacenarán** en una base de datos única y consistente, de manera que la definición de tangibles o proyectos / conceptos sea única. Ciertos campos, como este tipo de conceptos, solo deben poder ser definidos por el coordinador de área y los de proyectos, que actúan a modo de super usuarios.
- Debe **replicarse** el **sistema de registro de tareas** que ya presenta el sistema actual añadiendo campos como el tangible asociado o una casilla para indicar si se trata de un evento o reunión
- Los **colores** principalmente empleados en la aplicación serán los corporativos de la empresa, que son los siguientes:



Pantone 432 C

C = 65 R = 66 Hexadecimal
M = 43 G = 74 424A52
Y = 26 B = 82
K = 78



Pantone 131 C

C = 2 R = 199 Hexadecimal
M = 41 G = 148 C7940D
Y = 100 B = 13
K = 10

- La aplicación debe reservar (Aunque no lo implemente de manera efectiva, ya que depende de decisiones del equipo de trabajo) un espacio para mostrar **indicadores** tanto estadísticos como de rendimiento de cada persona.
- **El acceso y uso de los datos** es fundamental, y la herramienta debe implementar una base de datos relacional consistente tipo Mysql (o similar), por la siguientes razones:
 - o Estructura de datos clara, con entidades como los tangibles que pueden ser compartidos por varias personas, y atributos, como las horas de realización de una tarea, etc
 - o Se necesita una integridad de los datos para que todas las tareas se almacenen correctamente
 - o La estructura debe ser fácilmente escalable si finalmente se decide extender su uso a otras áreas o recursos
- La interfaz gráfica de la aplicación debe ser atractiva para fomentar la adopción en toda la empresa y garantizar la usabilidad

3. Análisis de la solución

3.1. Análisis de las posibles soluciones

Se van a listar las posibles soluciones que resolverían los requerimientos de la aplicación, para lo cual se ha tenido en cuenta la experiencia en distintos lenguajes del alumno que desarrolla e implementa la aplicación:

1. Desarrollo de toda la aplicación en Python puro, empleando como librerías frontend Flet (Librería más reciente pero sencilla y muy declarativa), PyQt / Pyside o incluso Tkinter (Aunque en ésta el alumno no tiene experiencia). Si se pretendiera que la aplicación fuera web (Que, a fin de cuentas, sería multiplataforma en el sentido que de que podría accederse desde un terminal móvil sin problema), se podría combinar un entorno de desarrollo como Django o Flask con el uso de lenguajes frontend tipo javascript, combinado con HTML y CSS.
2. Desarrollo de backend en java y frontend en Python: aunque es poco común, combinando librerías de interfaz gráfica de Python con la gestión de las solicitudes llevadas a cabo por el propio frontend, se harían las peticiones a la API de Springboot, siendo éste último quien gestione las peticiones a la base de datos.
3. Combinando java para el backend y javascript para el frontend. Uno de los enfoques más comunes, empleando Springboot para el backend y frameworks de referencia como React o Angular para el frontend. La aplicación podría incluso desarrollarse fullstack en javascript mediante Nodejs.
4. Desarrollo backend en java y frontend con flutter(Dart). Esta opción también es cada vez más común dado la capacidad que tiene el framework Flutter para compilar a iOS y Android, así como para ejecutarlo en navegadores web, asegurando una aplicación realmente multiplataforma y aprovechando lo completo que es el propio Flutter.

3.2. Evaluación de las posibles soluciones

A continuación, se muestra el DAFO de cada una de las cuatro soluciones propuestas:

	Opcion 1	Opción 2	Opción 3	Opción 4	Opcion 1	Opción 2	Opción 3	Opción 4	
Frontend	Python (Flet, PyQt...)	Python	Javascript	Flutter	Python (Flet, PyQt...)	Python	Javascript	Flutter	Frontend
Backend	Python	Java (Springboot)	Java (Springboot)	Java (Springboot)	Python	Java (Springboot)	Java (Springboot)	Java (Springboot)	Backend
DEBILIDADES					FORTALEZAS				
Sobre todo las librerías frontend de Python tienen menos soporte y están menos consolidadas, ademas de que son algo más limitadas frente a, por ejemplo, Flutter	No suele ser el uso de estos lenguajes y la integración entre ambos puede dar problemas tanto en desarrollo como en la producción. El alumno no tiene experiencia en integración de este tipo tan poco asistida	El conocimiento del alumno de javascript es limitado y la gestión asíncrona de peticiones puede ser compleja	Es de esperar actualizaciones progresivas de Flutter, lo que podría llevar a la necesidad de actualizar la aplicación en el futuro próximo	El alumno tiene experiencia en Python y aprender a usar nuevas librerías, unido a la sintaxis sencilla del lenguaje, puede ser más fácil	La sencillez de Python unido a la potencia de Java los hacen, por separado, elecciones con mucho sentido, a lo que se une la experiencia del alumno en estos dos lenguajes	Es una de las opciones más comunes y tanto el uso de la parte web de algún framework de referencia como el backend con Springboot tendrán una buena integración	Ambas son opciones sólidas con grandes ventajas en el uso de los respectivos Frameworks, como la inclusión de Material Design en Flutter o la posibilidad de usar Lombok o la gestión de usuarios en Springboot		
AMENAZAS					OPORTUNIDADES				
Python es más lento que otros lenguajes como Java	Las limitaciones de las librerías de Python en GUI pueden hacer más complejo el desarrollo de ciertas funciones o directamente limitar alguna de las funcionalidades según la complejidad final de la aplicación	El aprendizaje de uso de React o similar puede ser complejo en el poco tiempo que se prevé para desarrollar el trabajo	El conocimiento de Dart del alumno es limitado	Python tiene multitud de librerías muy interesantes, como pandas, matplotlib o datapane, que para la parte de indicadores y análisis pueden dar un gran valor añadido	El reto de integrar ambos podría suponer una manera de consolidar el uso de ambos lenguajes, particularmente de Python, con el que se requerirá un alto grado de implementación	Existen multitud de cursos y materiales gratuitos y accesibles para el soporte de estos dos tipos de librerías, dado que es una configuración bastante usual	El hecho de desarrollar en Flutter, de Google, unido al creciente grado de adopción (Y demanda de profesionales) asegura que el framework va a seguir siendo referencia y va a recibir mejoras y actualizaciones en el futuro, además de suponer una consolidación de experiencia relevante para el alumno		

3.3. Descripción de la solución escogida

Finalmente se decide escoger la Opción 4, por la experiencia del alumno (Que ya ha realizado algunas aplicaciones sencillas tanto con Springboot como con Flutter por separado durante el curso) así como por la oportunidad de profundizar en dos utilidades de alta demanda actualmente, tanto para programadores frontend, como backend y fullstack.

Como fortalezas de **Springboot**; el uso de lombok, hibernate, JPA y demás automatizaciones, la facilidad de programar las operaciones CRUD de interacción con la base de datos o la gestión de usuarios son algunos de los aspectos diferenciadores.

Como fortalezas de **Flutter**; su carácter multiplataforma, la potencia de Dart como lenguaje, la disponibilidad de widgets y de la biblioteca Material Design de Google (Implementada mediante material.dart), y la cantidad de documentación disponible.

Como resumen, la solución se compondrá de los siguientes elementos:

- **Base de datos SQL**, donde se almacenará toda la información sobre usuarios, tareas, estado de las tareas, indicadores, tangibles, proyectos, etc. Para desarrollarla se empleará MySQL workbench y para testearla se empleará previsiblemente Docker.
- **Backend Java mediante Springboot**, creando así un proyecto tipo Spring donde gestionaremos los accesos a los datos, la seguridad, los usuarios y la API REST. Para desarrollarlo y testearlo se empleará Spring Tool suite.
- **Frontend Dart mediante Flutter**, implementando toda la interfaz de usuario, navegación, y las peticiones a la API REST de Java.

4. Diseño de la solución.

4.1. Requisitos.

Como ya se introducía previamente, la aplicación tiene dos requisitos funcionales principales:

- Permitir un seguimiento efectivo de las **tareas** diarias que cada recurso se propone realizar en un contexto de ejecución dentro de **proyectos** con **tangibles** (resultados, objetivos materializados) específicos
- Realizar un análisis de **rendimiento, desviaciones** y otros **indicadores** de interés

Estos requisitos se traducen, a su vez, en una serie de requisitos de detalle que la herramienta debe cumplir:

Requisito 1. Permitir un seguimiento efectivo de las **tareas** diarias que cada recurso se propone realizar en un contexto de ejecución dentro de **proyectos** con **tangibles** (resultados, objetivos materializados) específicos

- **Registrar o eliminar tareas** diarias a realizar. Cada tarea será única de cada usuario con el fin de evitar registros de tareas demasiado extensos sobre los que seleccionar
- Asociar a cada tarea una cantidad de **horas prevista** y una cantidad de **horas dedicadas**, y en base a la planificación global del día, proporcionar información sobre la sobre o infraasignación del recurso según la **capacidad diaria** que éste indique
- Permitir, en la definición de cada tarea, que ésta sea asociada a un **proyecto** (P.ej. Proyecto LAMBDA¹, Proyecto NEMESIS², Oportunidades) y a un **tangible** (P.ej. Informe de Inicio, modelo de correlación). Ambas entidades (Proyecto y tangible) deben tener instancias compartidas, es decir, tanto a cada proyecto como a cada tangible, los usuarios deberán seleccionar sobre una En este sentido, por seguridad, se definen un par de aclaraciones:
 - Los proyectos serán dados de alta en el sistema por el coordinador de área. Cuando un proyecto sea dado de alta no podrá ser eliminado salvo tras confirmación expresa y reiterada de un usuario con privilegios necesarios

¹ Nombre ficticio

² Nombre ficticio

- Durante el día en curso, el usuario podrá **añadir tareas** sin problema, pero nunca podrá dar de alta tareas “a pasado”. Cuando se dé de alta una tarea se ha de indicar la descripción, el proyecto y el tangible al que se asocia. Se pueden prever otros campos, como por ejemplo indicar si la tarea es una reunión en sí misma, con el fin de poder hacer posteriores análisis del porcentaje de tiempo dedicado a reuniones, por poner un ejemplo.
- Al finalizar cada día, a las 23:59, **las tareas de dicho día quedarán bloqueadas** sin poder modificarse, de manera que solo podrá acceder en “modo visualización” para consultar los datos que dio de alta cada día pasado
- Para cada tarea, se generará un **valor de desviación** entre el tiempo que inicialmente se tenía previsto para la tarea y el tiempo finalmente dedicado. Se debe valorar incluir asimismo campos que permitan afinar mejor el seguimiento y la representatividad de las tareas que se registran, como por ejemplo un campo de “tarea no prevista” o un campo de “tarea finalizada”, que sean simplemente indicativos pero permitan hacer un mejor análisis de los datos.

Requisito 2. - *Realizar un análisis de rendimiento, desviaciones y otros indicadores de interés*

Parte de ese análisis del rendimiento ya se lleva a cabo con el valor de desviación que se indica para cada tarea, indicador que ya proporciona información sobre lo bien que planifica cada usuario sus tareas. Partiendo de ese punto, se debe contemplar que el sistema cumpla los siguientes requerimientos:

- Proporcionar un **análisis básico de grado de rendimiento individual** a cada usuario de la aplicación en base a sus tareas como individuo. Para ello, se deben realizar algunos cálculos estadísticos básicos (Ya que los framework escogidos tampoco son especialmente fuertes en análisis y representación avanzada de datos). Algunos de los indicadores que se propone calcular son:
 - Cantidad de tareas mensuales completadas
 - % Mensual de cumplimiento de tareas
 - Cantidad de tareas no previstas
 - % de tiempo mensual de tareas no previstas

Estos indicadores podrán ser, no obstante, revisados según la complejidad de implementación y dificultad de obtención de datos, e incluso ampliados si se detectan mejoras o métricas específicas de mayor interés. No obstante, todos los indicadores se

calcularán a partir del marco de datos (Cantidad y tipo de datos) especificados en el requisito 1.

- Proporcionar un **análisis general de rendimiento por recurso**, con indicadores de seguimiento general tanto a nivel de todos los recursos (Datos agregados) como a nivel de cada usuario de la aplicación (Rendimiento individual). Estos datos serán de interés, sobre todo, para el Coordinador de área, que será el único que tenga acceso a ellos. Se podrán designar varios coordinadores de área.
- Proporcionar un **análisis general de rendimiento por recurso y proyecto**. De manera similar al anterior pero acotado a todos los recursos de un proyecto dado, siendo información accesible solo para el Coordinador de proyecto. Se podrán designar varios coordinadores de proyecto.

Estos requisitos, no obstante, se subordinarán totalmente a la consecución del Requisito 1, que garantiza la funcionalidad de la aplicación de manera general, además de estar condicionados a la aprobación o propuesta de indicadores por parte del equipo de trabajo de la empresa según sus intereses y la posible alineación con indicadores institucionales.

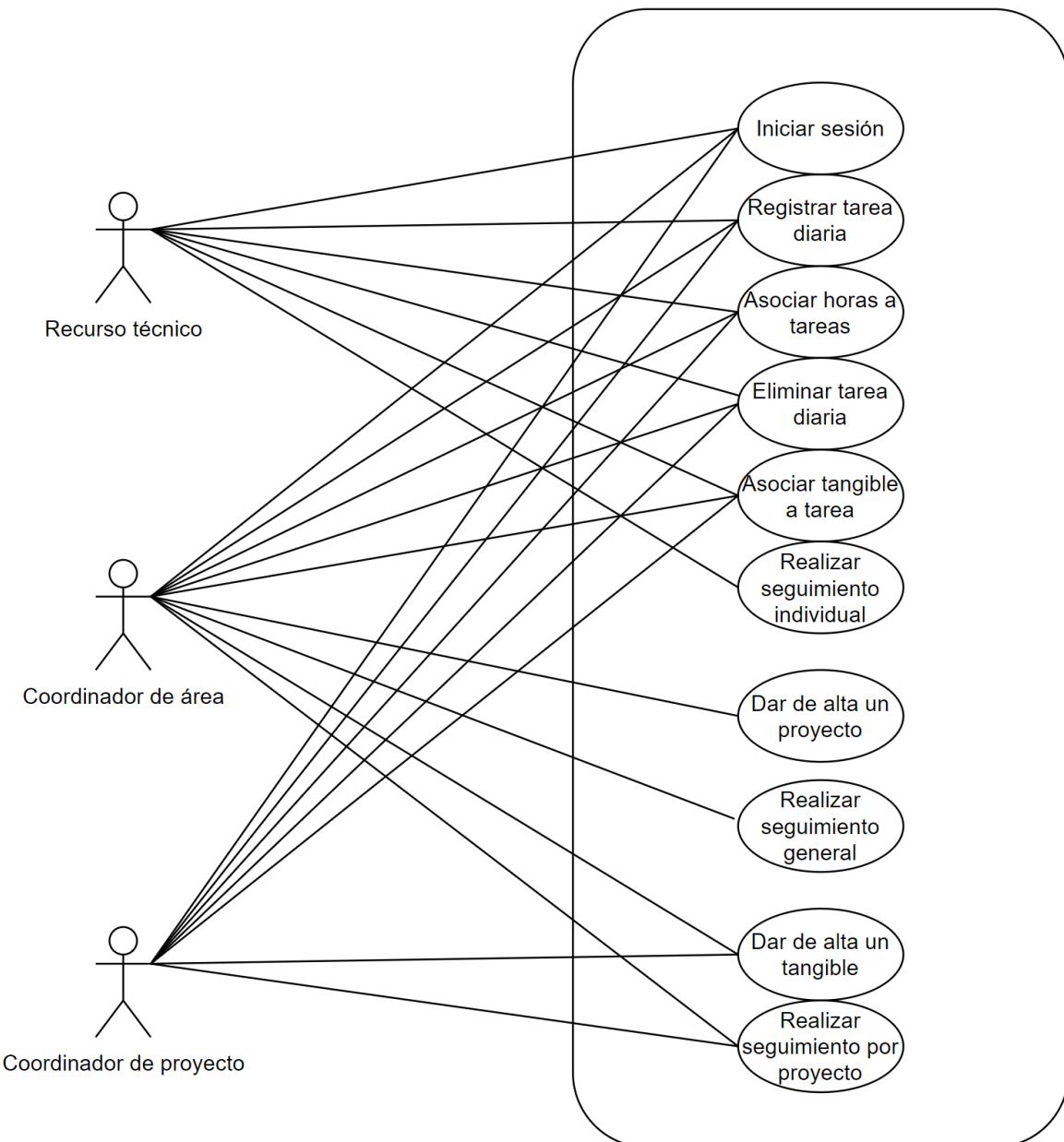
4.2. Análisis de riesgos.

A continuación, se exponen los principales riesgos detectados, clasificados por nivel de importancia, desde alto riesgo hasta bajo riesgo:

Riesgo	Descripción	Nivel de importancia
Complejidad de la lógica de cálculo	En lo que vendría a definirse como "lógica de negocio" para calcular algunos indicadores es probable que haya que hacer análisis con orientación ligeramente estadística de un gran número de tareas. La complejidad de esta lógica puede determinar la correcta funcionalidad de la aplicación.	Medio. La lógica debe ser definida de manera sencilla incluso en detrimento de no poder calcular algunos indicadores, ya que gran parte de la consistencia de la aplicación está asociada a la capacidad de registro y seguimiento y no tanto a la explotación de los datos de rendimiento.
Desbordamiento de tareas del sistema	Dado que cada usuario tiene un número potencialmente infinito de tareas que poder registrar (Las tareas se las genera cada usuario según su criterio propio), se pueden dar casos en que el sistema tenga una gran cantidad de tareas registradas si se emplea durante mucho tiempo por parte de muchos usuarios.	Bajo. La cantidad de usuarios prevista inicialmente (8-9) no hace pensar que en el medio plazo vaya a haber una cantidad excesiva de tareas registradas en sistema
Diseño y estructuración incorrecta de la base de datos	Un diseño incorrecto de la base de datos, que contiene gran parte del valor añadido de la aplicación (Cuyas funcionalidades principales se orientan a registrar correctamente y dar seguimiento a esos datos) puede llevar a que la aplicación no funcione todo lo bien que se espera.	Medio. El diseño debe considerar el uso final de la aplicación y la base de datos, relaciones entre entidades (Usuarios, proyectos, tangibles, tareas...) deben ser consistentes y coherentes a la necesidad
Desarrollo excesivo	Es posible que el tiempo de desarrollo sea mayor del previsto inicialmente, tanto por la falta de práctica del alumno en el desarrollo fullstack de aplicaciones, como por la falta de experiencia en la puesta en producción de estas.	Alto. Debe considerarse como escenario realista implementar sobre todo la primera funcionalidad prevista para la aplicación (Registro y visualización de tareas). Según el grado de avance, se ajustará el alcance de implementación de la segunda (KPIs y seguimiento mensual)

4.3. Casos de uso.

A continuación, se muestra el diagrama de casos de uso:



4.4. Diagramas

A continuación, se muestran algunos diagramas y documentación gráfica de ayuda. El primero de ellos es el prototipo de baja-media fidelidad sobre el que se orienta el desarrollo de la interfaz de usuario:

Pantalla 1: Login



Pantalla 2: Tareas

Seguimiento diario
○ ○ ○

◀ ▶ C
◀ ▶ C

Tareas
22/04/2024
📅
Alejandro Rubio Rico

KPIs
Capacidad
5
Capacidad vs plan
0
5
5.5
+0.5

Proyecto
Tangible
Tarea
¿R/E?
Planificadas
Dedicadas
Desviación

Proyecto	Tangible	Tarea	¿R/E?	Planificadas	Dedicadas	Desviación
LAMBDA	Especificaciones	Definición KPIs con Belén	X	2	1.5	-0.5
LAMBDA	Informe final	Desarrollo informe		3	4	+1

Seguimiento diario

22/04/2024

Alejandro Rubio Rico

Tareas

KPIs

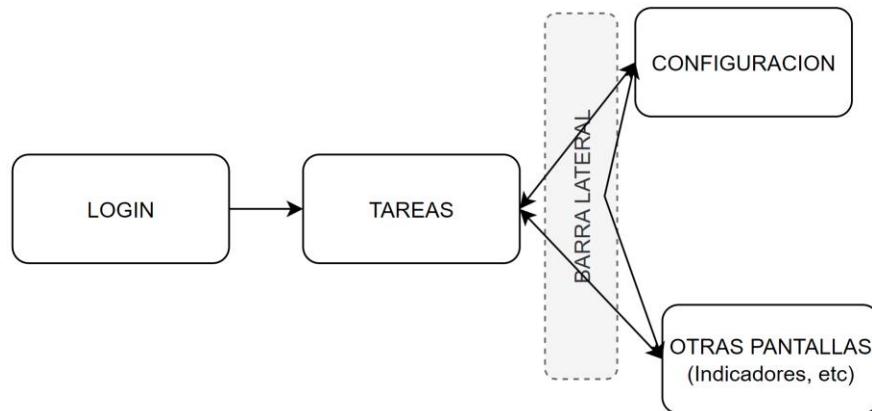
Config.

< Noviembre 2024 >

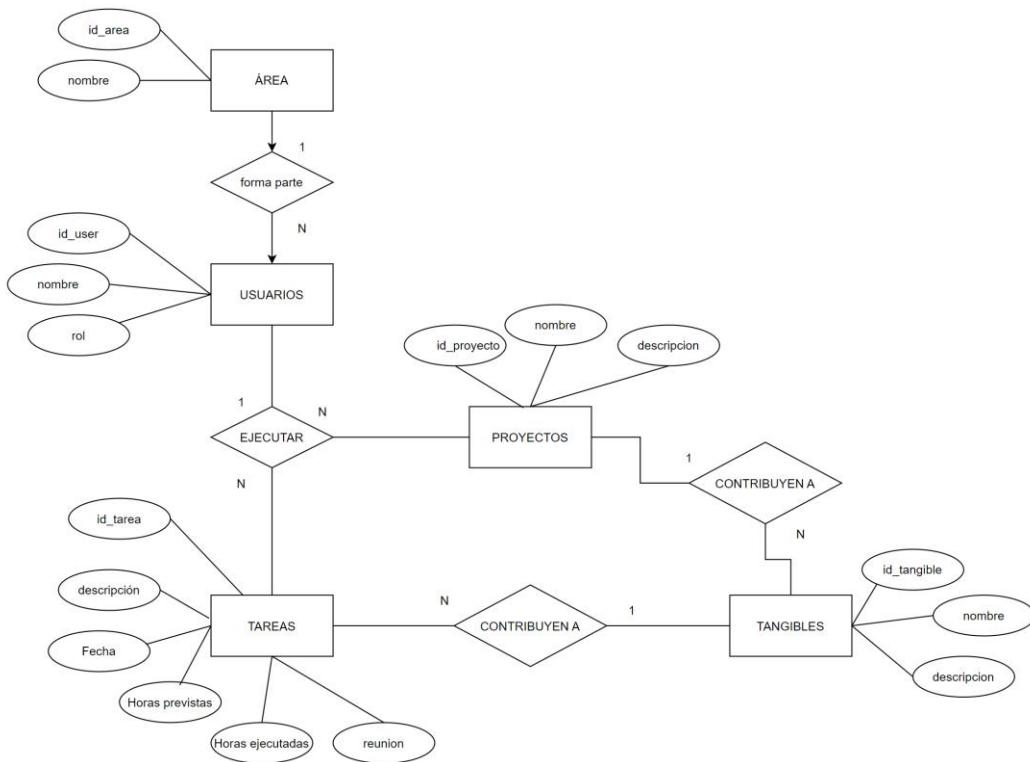
SUN	MON	TUE	WED	THU	FRI	SAT
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

0	5	5.5	+0.5
¿R/E?	Planificadas	Dedicadas	Desviación
X	2	1.5	-0.5
	3	4	+1

A continuación, se muestra un diagrama de navegación. Como se aprecia, es muy sencillo ya que gran parte de la lógica de la aplicación se concentra en la propia pantalla de tareas, que es además el punto de entrada a la aplicación. La navegación entre pantallas se realiza mediante la barra lateral:



Por otro lado, se adjunta el diagrama Entidad Relación.



5. Desarrollo de la solución

5.1. Estimación de coste temporal

Para la estimación del coste temporal y económico asociado a la implementación se ha realizado un diagrama de Gantt con asignación de recursos:

TAREAS	DETALLE	RECURSOS IMPLICADOS	Días																														
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	Vida útil app				
Planificación																																	
Requisitos y análisis de viabilidad	Recolección de información, entrevistas, documentación sistema actual, evaluación de recursos, análisis de riesgos	PO, Dev, LM*																															
Diseño de la arquitectura	Diseño de la arquitectura y selección de tecnologías y herramientas a emplear	DB, DF																															
Diseño																																	
Diseño base de datos	Modelado de datos y tablas, diagramas ER	DB																															
Diseño backend	Diseño de estructura backend y modelos de datos	DB																															
Diseño frontend	Diseño de interfaces, prototipado, definición de la navegación y usabilidad	DI, DF, PO																															
Desarrollo backend																																	
Desarrollo base de datos	Creación estructura de base de datos, testeo	DB																															
Desarrollo base springboot	Implementación DTOs, controladores, servicios, repositorio	DB																															
Conexión base de datos	Configuración conexión Mysql, pruebas	DB																															
Implementación lógica de negocio	Adaptación de controladores a necesidades	DB, DF																															
Desarrollo frontend																																	
Implementación de interfaz general	Desarrollo login y estructura general, barra de navegación, botones comunes, aspecto estético	DI, DF																															
Desarrollo pantallas	Desarrollo pantallas aplicación, desarrollo widgets, ubicación placeholders	DF																															
Desarrollo controladores y servicios	Desarrollo lógica de datos de la aplicación, funcionalidades adicionales	DF, DB																															
Integración con backend	Programación API, envío y recepción de datos, adaptación de formatos	DF, DB																															
Implementación y pruebas																																	
Pruebas unitarias	Pruebas funcionales backend, frontend	DB, DF, QA																															
Pruebas de integración y seguridad	Pruebas API, testeo vulnerabilidades, carga, rendimiento	DB, DF, QA																															
Pruebas de usuario final	Pruebas generales de cara a uso de usuario final	QA, PO																															
Desarrollo de documentación	Desarrollo de detalle de la documentación de la aplicación, tanto de usuario como del código	DB, DF																															
Despliegue y seguimiento																																	
Despliegue en producción	Despliegue on-premise o cloud, publicación app cliente en google play	DO																															
Seguimiento	Monitoreo logs, bugs, actualizaciones, seguimiento	DO, SP																															
LEYENDA			Coste salarial estimado				Dedición total (d)				Dedición total (h)				Coste total				Lead time														
Product owner	PO	45 €/h	7	56 h	2.520,00 €	26 días																											
Lead manager	LM	40 €/h	4	35,2 h	1.408,00 €																												
Diseñador UI/UX	DI	30 €/h	3	24 h	720,00 €																												
Desarrollador backend / fullstack	DB	35 €/h	26	104 h	3.640,00 €																												
Desarrollador frontend / fullstack	DF	35 €/h	30	120 h	4.200,00 €																												
Devops	DO	40 €/h	3	24 h	960,00 €																												
Quality Analysts / Testers	QA	25 €/h	5	40 h	1.000,00 €																												
Soporte	SP	20 €/h	5	40 h	800,00 €																												
* En el resto de tareas, el Lead Manager o Lead Developer llevará un seguimiento del proyecto a nivel de gestión			15.248,00 € *																														
* Excluido mantenimiento																																	

* En el resto de tareas, el Lead Manager o Lead Developer llevará un seguimiento del proyecto a nivel de gestión

* Excluido mantenimiento

5.2. Dotación de recursos

Los recursos de los que se dispone (tanto humanos como materiales) son:

- Desarrollador fullstack (Considerado como tal el propio alumno que presenta este trabajo, si bien el perfil es más bien de desarrollador backend)
- Horas de asistencia técnica de recursos del departamento de Sistemas informáticos y Ciberseguridad de la empresa (Cantidad limitada)
- Prototipo / ejemplo de programación completo API REST en Springboot desarrollado en módulo de Acceso a Datos
- Equipo de Desarrollo PC ASUS TUF Intel Core i7-12700H 2.3 GHz 32 GB RAM Windows 11 Pro. Monitor Samsung para apoyo en el desarrollo.
- Como IDE, se han empleado MySQL Workbench (Community Edition), SpringToolSuite y Visual Studio Code con las extensiones pertinentes para el desarrollo Flutter
- Como herramienta de virtualización de la base de datos se ha empleado Docker a través de Docker Desktop
- Como herramienta de edición y control de cambios y versiones, se ha empleado Git a través de Github

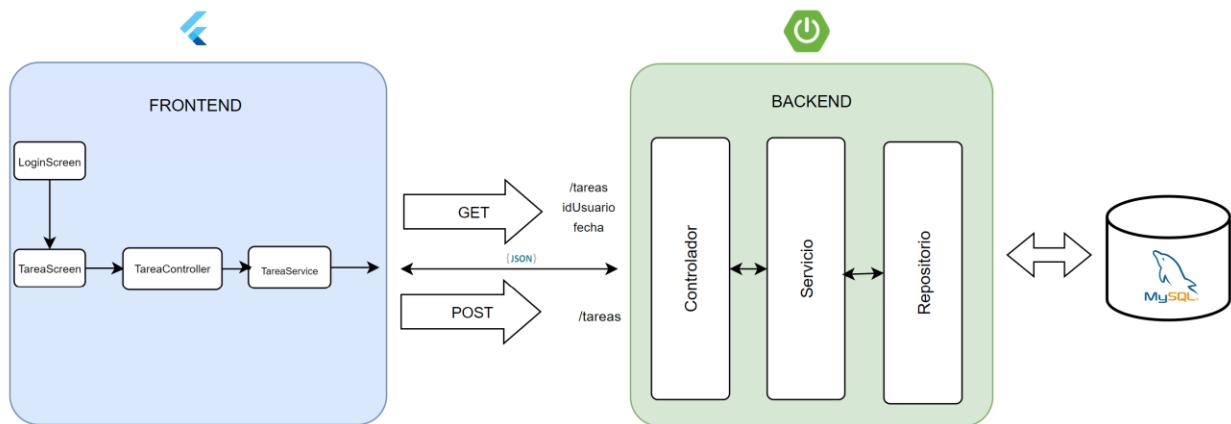
Además, se ha contado con el asesoramiento de la tutora individual para la resolución de dudas sobre enfoque, alcance y resolución del proyecto.

5.3. Desarrollo del sistema

El desarrollo del sistema se ha estructurado en los tres elementos principales que ya se indicaban previamente en la memoria:

- La base de datos SQL
- El backend desarrollado en Springboot, que implementa una API REST
- El frontend desarrollado en Flutter como framework multiplataforma de referencia

La siguiente imagen muestra una simplificación de la arquitectura con los componentes principales:



El desarrollo sigue la secuencia descrita en el apartado 5.1, comenzando con el desarrollo y testeo de la base de datos, continuando con un planteamiento básico del backend hasta el punto de implementar los controladores. En paralelo, se ha desarrollado el diseño de la aplicación desde el punto de vista de interfaz y experiencia de usuario (UX/UI). En el momento en que los controladores de la API están preparados para ser implementados, se comienza un desarrollo paralelo para testeo cruzado de envío y recepción de datos, culminando con el desarrollo de la aplicación hasta el alcance seleccionado.

A continuación, se detalla cada una de las partes del desarrollo y sus resultados.

Desarrollo de la base de datos

Para la implementación de la base de datos se ha desarrollado un script MySQL empleando la versión Community de Mysql workbench.

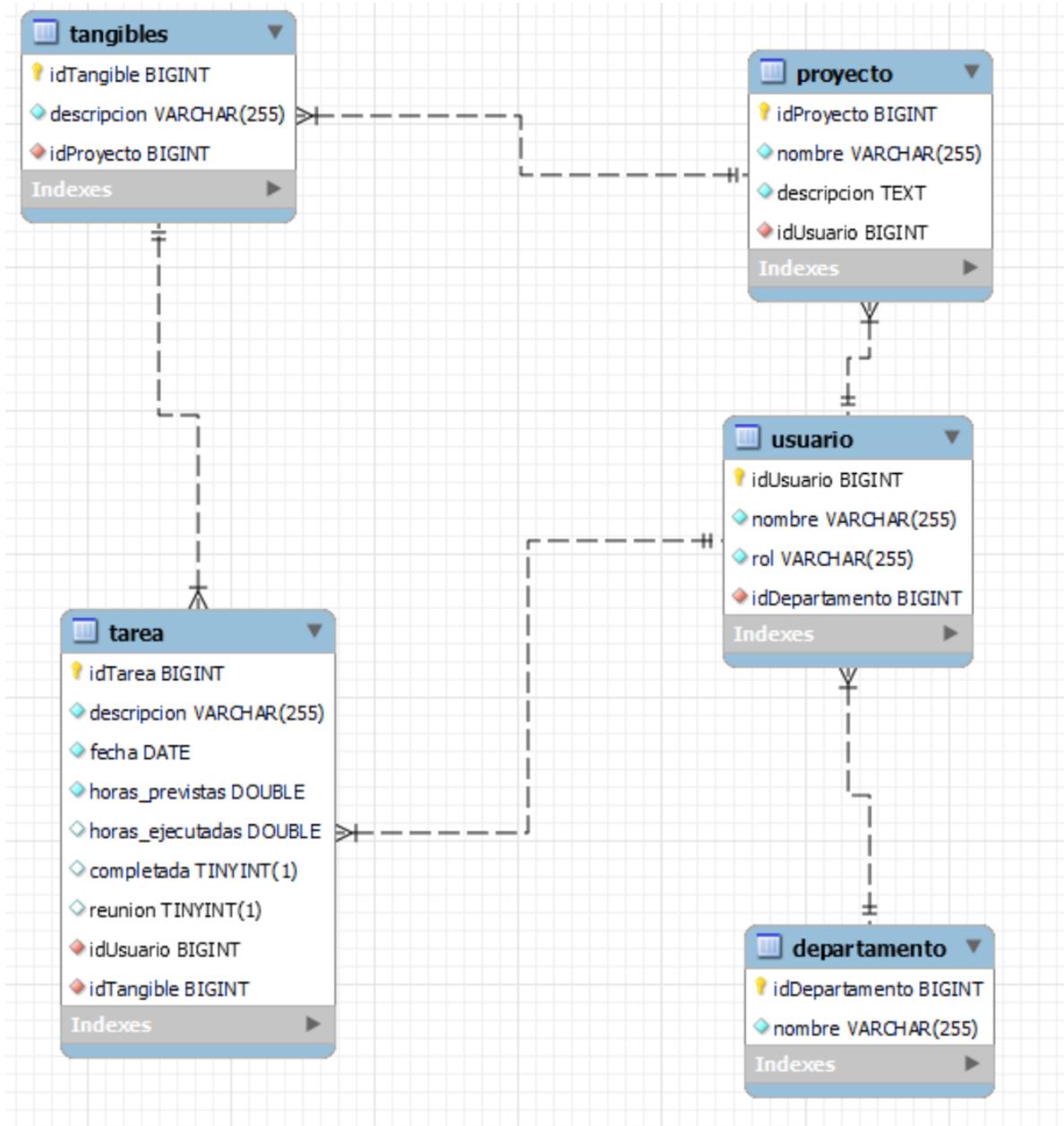


```

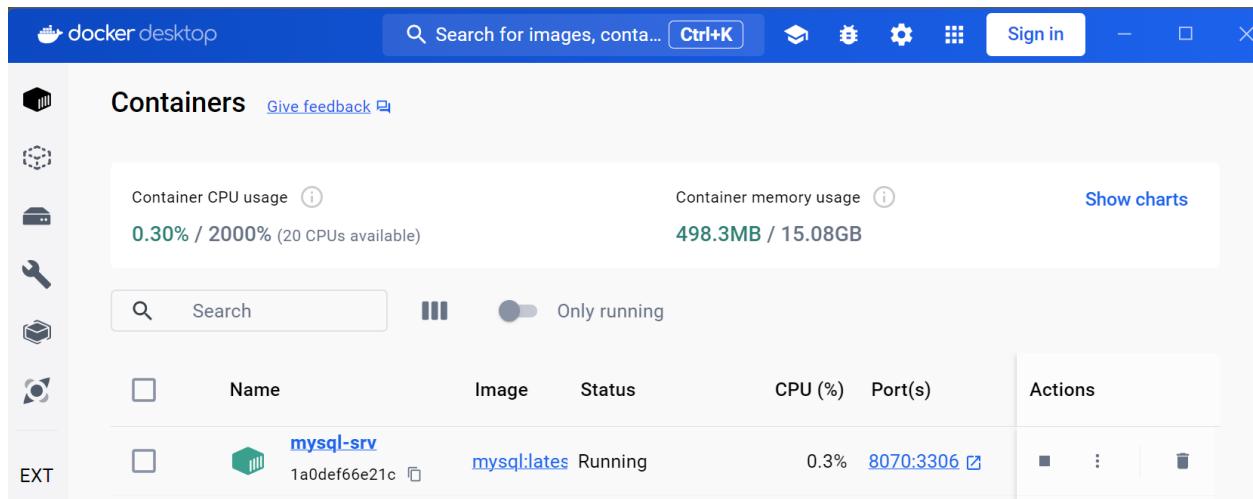
BD_tareas x
1 -- BASE DE DATOS PARA APLICACIÓN DE SEGUIMIENTO DE TAREAS --
2
3 ● DROP DATABASE IF EXISTS tareas;
4
5 -- BASE DE DATOS PARA APLICACIÓN DE SEGUIMIENTO DE TAREAS --
6
7 ● DROP DATABASE IF EXISTS tareas;
8
9 ● CREATE DATABASE tareas;
10
11 ● USE tareas;
12
13 -- Creación de la tabla departamento
14 ● ○ CREATE TABLE departamento (
15     idDepartamento BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
16     nombre VARCHAR(255) NOT NULL
17 );
18
19 -- Creación de la tabla usuario
20 ● ○ CREATE TABLE usuario (
21     idUsuario BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
22     nombre VARCHAR(255) NOT NULL,
23     rol VARCHAR(255) NOT NULL,
24     idDepartamento BIGINT NOT NULL,
25     FOREIGN KEY (idDepartamento) REFERENCES departamento(idDepartamento)
26 );
27
28 -- Creación de la tabla proyecto
29 ● ○ CREATE TABLE proyecto (
30     idProyecto BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
31     nombre VARCHAR(255) NOT NULL,
32     descripcion TEXT NOT NULL,
33     idUsuario BIGINT NOT NULL,
34     FOREIGN KEY (idUsuario) REFERENCES usuario(idUsuario)
35 );
36
37 -- Creación de la tabla tangibles
38 ● ○ CREATE TABLE tangibles (
39     idTangible BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
40     descripcion VARCHAR(255) NOT NULL,
41     idProyecto BIGINT NOT NULL,
42     FOREIGN KEY (idProyecto) REFERENCES proyecto(idProyecto)

```

A continuación, se adjunta captura del esquema de la base de datos implementada (Diagrama Entidad Relación proporcionado por MySQL):



La base de datos se ha montado en un contenedor Docker en el puerto 8070 en localhost durante el tiempo de desarrollo:



La base de datos ha sido dada de alta asimismo con algunas tareas y otros datos ficticios con el fin de verificar su integridad y posibilitar la realización de pruebas:

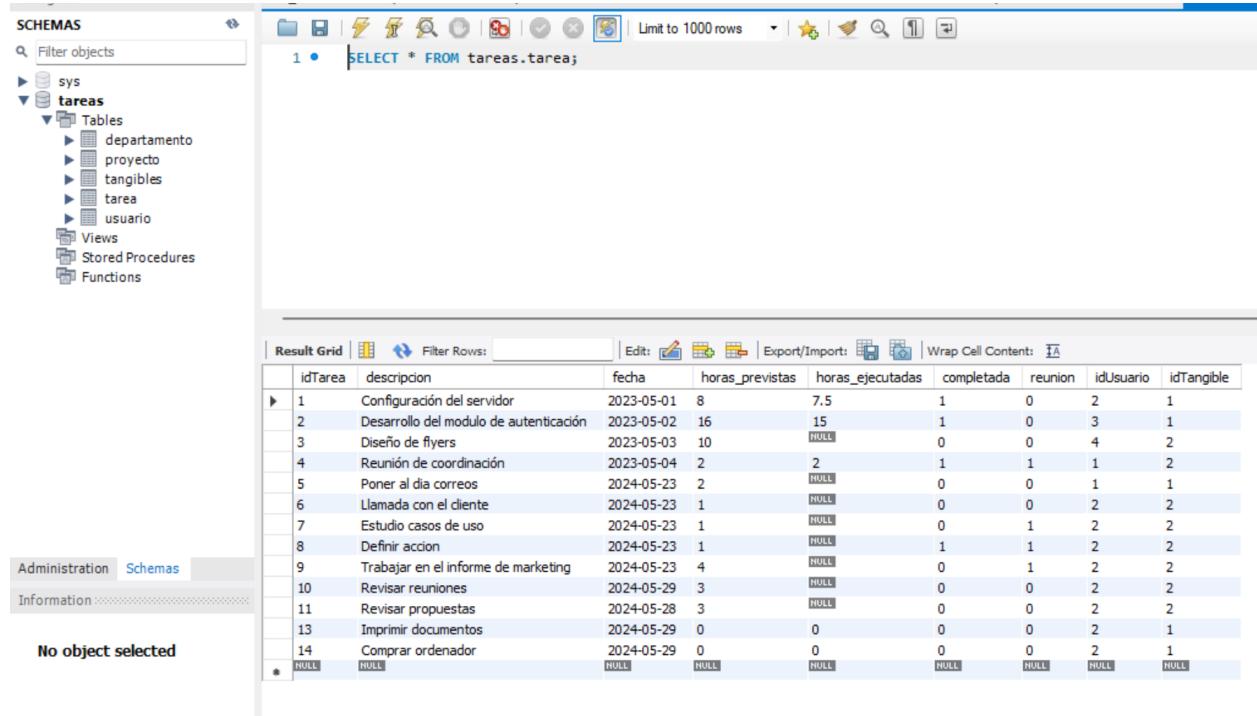
```
-- Proyectos
INSERT INTO proyecto (nombre, descripcion, idUsuario) VALUES ('Proyecto Alpha', 'Proyecto Alpha', 3);
INSERT INTO proyecto (nombre, descripcion, idUsuario) VALUES ('Proyecto Beta', 'Proyecto Beta', 2);

-- tangibles
INSERT INTO tangible (descripcion, idProyecto) VALUES ('Servidor de desarrollo', 1);
INSERT INTO tangible (descripcion, idProyecto) VALUES ('Material de publicidad', 2);

-- Insertar tareas
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Configuración del servidor', '2023-05-01', 8.0, 7.5, TRUE, FALSE, 2, 1);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Desarrollo del módulo de autenticación', '2023-05-02', 16.0, 15.0, TRUE, FALSE, 3, 1);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Diseño de flyers', '2023-05-03', 10.0, NULL, FALSE, FALSE, 4, 2);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Reunión de coordinación', '2023-05-04', 2.0, 2.0, TRUE, TRUE, 1, 2);

INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Poner al día correos', '2024-05-23', 2.0, NULL, FALSE, FALSE, 1, 1);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Llamada con el cliente', '2024-05-23', 1.0, NULL, FALSE, FALSE, 2, 2);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Estudio casos de uso', '2024-05-23', 1.0, NULL, FALSE, TRUE, 2, 2);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Definir acción', '2024-05-23', 1.0, NULL, TRUE, TRUE, 2, 2);
```

Como se aprecia en la siguiente captura, la base de datos ha sido creada y dada de alta satisfactoriamente y es funcional:



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view shows the 'tareas' schema with tables like departamento, proyecto, tangibles, tarea, and usuario. A query window at the top displays the results of the SQL command: 'SELECT * FROM tareas.tarea;'. Below it, the 'Result Grid' shows a list of tasks with columns: idTarea, descripción, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, and idTangible. The data includes various tasks such as 'Configuración del servidor', 'Desarrollo del modulo de autenticación', and 'Reunión de coordinación', each with specific details like date, estimated hours, and completion status.

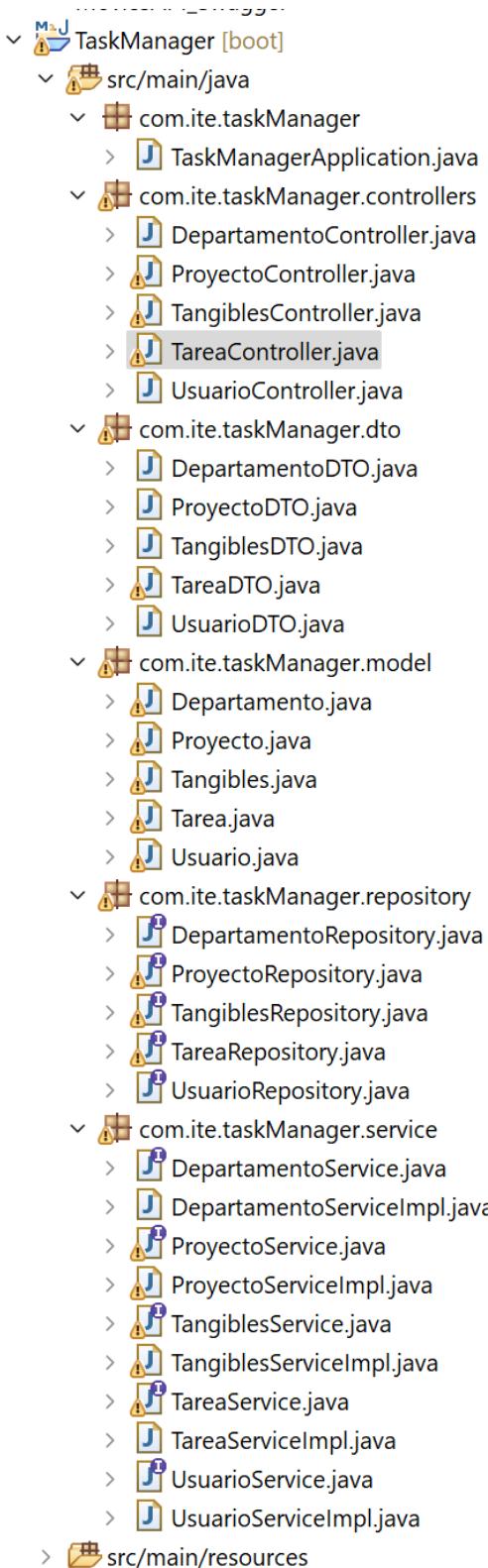
idTarea	descripción	fecha	horas_previstas	horas_ejecutadas	completada	reunion	idUsuario	idTangible
1	Configuración del servidor	2023-05-01	8	7.5	1	0	2	1
2	Desarrollo del modulo de autenticación	2023-05-02	16	15	1	0	3	1
3	Diseño de flyers	2023-05-03	10	NULL	0	0	4	2
4	Reunión de coordinación	2023-05-04	2	2	1	1	1	2
5	Poner al dia correos	2024-05-23	2	NULL	0	0	1	1
6	Llamada con el cliente	2024-05-23	1	NULL	0	0	2	2
7	Estudio casos de uso	2024-05-23	1	NULL	0	1	2	2
8	Definir acción	2024-05-23	1	NULL	1	1	2	2
9	Trabajar en el informe de marketing	2024-05-23	4	NULL	0	1	2	2
10	Revisar reuniones	2024-05-29	3	NULL	0	0	2	2
11	Revisar propuestas	2024-05-28	3	NULL	0	0	2	2
13	Imprimir documentos	2024-05-29	0	0	0	0	2	1
14	Comprar ordenador	2024-05-29	0	0	0	0	2	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Desarrollo del backend

El backend, como ya se adelantaba, ha sido implementado enteramente en Springboot. Dada la potencia del framework, se ha seguido estrictamente la estructura por capas de éste, lo que a pesar de ser algo más trabajoso (Sobre todo a la hora de generar las distintas clases de cada uno de los paquetes para todas las entidades) posibilita una escalabilidad y mejora el mantenimiento del sistema. La siguiente figura refleja dicha arquitectura:



La aplicación, denominada TaskManager (Gestor de tareas) tiene la siguiente estructura de contenidos:



Una de las clases principales es Tarea, ya que agrupa gran parte de las características importantes y suponen la entidad principal de la aplicación, que al fin y al cabo es una app de seguimiento de tareas. A continuación se muestra el DTO de Tareas (Que es el más complejo de los implementados):

```

@Data
public class TareaDTO implements Serializable {

    private static final long serialVersionUID = 18L;

    private Long idTarea;
    private String descripcion;
    private Date fecha;
    private Double horasPrevistas;
    private Double horasEjecutadas;
    private Boolean completada;
    private Boolean reunion;
    //private Usuario usuario;
    //private Tangible tangible;

    private Long usuarioId; // Solo el ID del usuario
    private Long tangibleId; // Solo el ID del tangible

    private String tangibleDescripcion;
    private String proyectoDescripcion;

    // De Modelo a DTO
    public static TareaDTO Model2DTO(Tarea tarea) {
        TareaDTO tareaDTO = new TareaDTO();
        tareaDTO.idTarea = tarea.getIdTarea();
        tareaDTO.descripcion = tarea.getDescripcion();
        tareaDTO.fecha = tarea.getFecha();
        tareaDTO.horasPrevistas = tarea.getHorasPrevistas();
        tareaDTO.horasEjecutadas = tarea.getHorasEjecutadas();
        tareaDTO.completada = tarea.getCompletada();
        tareaDTO.reunion = tarea.getReunion();
        //tareaDTO.usuario = tarea.getUsuario();
        //tareaDTO.tangible = tarea.getTangible();
        tareaDTO.usuarioId = tarea.getUsuario().getIdUsuario();
        tareaDTO.tangibleId = tarea.getTangible().getIdTangible();
        tareaDTO.tangibleDescripcion = tarea.getTangible().getDescripcion();
        //tareaDTO.proyectoDescripcion = tarea.getTangible().getProyecto().getDescripcion();

        return tareaDTO;
    }
}

```

Y el modelo de datos original:

```

package com.ite.taskManager.model;

import java.sql.Date;[]

@Data
@Entity
@Table(name = "tarea")
public class Tarea {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long idTarea;

    @Column(name = "descripcion", nullable = false)
    private String descripcion;

    @Column(name = "fecha")
    private Date fecha;

    @Column(name = "horas_previstas", nullable = false)
    private Double horasPrevistas;

    @Column(name = "horas_ejecutadas")
    private Double horasEjecutadas;

    @Column(name = "completada")
    private Boolean completada;

    @Column(name = "reunion")
    private Boolean reunion;

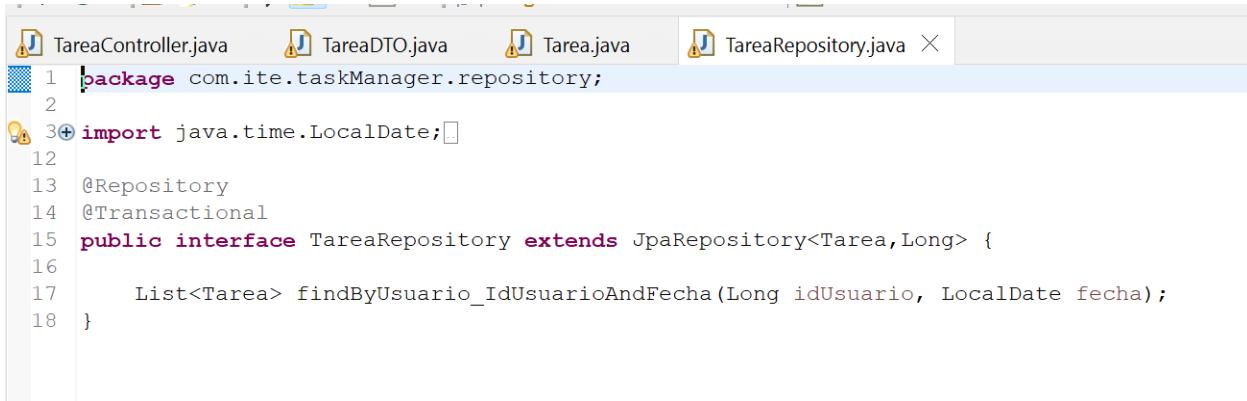
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "idUsuario")
    //@ToString.Exclude
    //@JsonBackReference
    @JsonIgnore
    private Usuario usuario;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "idTangible")
    //@ToString.Exclude
    //@JsonBackReference
    @JsonIgnore
    private Tangible tangible;
}

```

Como se aprecia, para evitar bucles recurrentes se ha empleado JsonIgnore, ya que a priori el frontend para las funcionalidades implementadas no necesita de objetos Usuario y Tangible completos (Y existen otras maneras de obtener estas características).

Se han implementado asimismo repositorios para todas las clases:



```

1 package com.ite.taskManager.repository;
2
3 import java.time.LocalDate;[]
4
5 @Repository
6 @Transactional
7 public interface TareaRepository extends JpaRepository<Tarea, Long> {
8
9     List<Tarea> findByUsuario_IdUsuarioAndFecha(Long idUsuario, LocalDate fecha);
10 }

```

Y servicios (En este caso, siguiendo el patrón *Fachada*):

The screenshot shows two Java code files in an IDE:

- TareaService.java**: An interface with methods for getting a task by ID, listing all tasks, finding tasks by user ID and date, and saving a task.
- TareaServiceImpl.java**: A implementation class that uses autowiring to inject `TareaRepository`, `UsuarioRepository`, and `TangiblesRepository`. It overrides the service methods to perform database queries and map results to DTOs.

```
1 package com.ite.taskManager.service;
2
3 import java.time.LocalDate;
4
5 public interface TareaService {
6     TareaDTO getTareaById(Long id);
7     List<TareaDTO> listAllTareas();
8     List<TareaDTO> findByUsuario_IdAndFecha(Long idUsuario, LocalDate fecha);
9     TareaDTO save(TareaDTO tareaDTO);
10 }
11
12
13
14
15
```

```
1 package com.ite.taskManager.service;
2
3 import java.time.LocalDate;
4
5 @Service
6 public class TareaServiceImpl implements TareaService {
7
8     @Autowired
9     TareaRepository tareaRepository;
10
11     @Autowired
12     UsuarioRepository usuarioRepository;
13
14     @Autowired
15     TangiblesRepository tangiblesRepository;
16
17     @Override
18     public TareaDTO getTareaById(Long id) {
19         Tarea tarea = tareaRepository.findById(id).orElse(null);
20         return tarea != null ? TareaDTO.Model2DTO(tarea) : null;
21     }
22
23     @Override
24     public List<TareaDTO> listAllTareas() {
25         List<Tarea> lasTareas = tareaRepository.findAll();
26         List<TareaDTO> lasTareasDTO = new ArrayList<>();
27
28         for (Tarea t : lasTareas) {
29             TareaDTO tareaDTO = TareaDTO.Model2DTO(t);
30             lasTareasDTO.add(tareaDTO);
31         }
32
33         return lasTareasDTO;
34     }
35
36     @Override
37     public List<TareaDTO> findByUsuario_IdAndFecha(Long idUsuario, LocalDate fecha) {
38         List<Tarea> tareas = tareaRepository.findByUsuario_IdUsuarioAndFecha(idUsuario, fecha);
39         return tareas.stream().map(TareaDTO::Model2DTO).collect(Collectors.toList());
40     }
41
42     @Override
43     public TareaDTO save(TareaDTO tareaDTO) {
44         // Convertir TareaDTO a entidad Tarea
45         Tarea tarea = new Tarea();
46         tarea.setDescripcion(tareaDTO.getDescripcion());
47
48         // tarea.setFecha(tareaDTO.getFecha());
49
50         // Verificar si se proporcionó una fecha en la solicitud
51
52         // Si no se proporciona una fecha, establecer la fecha actual
53
54         tarea.setFecha(Date.valueOf(LocalDate.now()));
55
56         tarea.setHorasPrevistas(tareaDTO.getHorasPrevistas());
57         tarea.setHorasEjecutadas(tareaDTO.getHorasEjecutadas());
58
59         tarea.setEstado(Tarea.ESTADO.PENDIENTE);
60
61         tareaRepository.save(tarea);
62
63         return TareaDTO.Model2DTO(tarea);
64     }
65
66     @Override
67     public void delete(TareaDTO tareaDTO) {
68         Tarea tarea = tareaRepository.findById(tareaDTO.getId()).orElse(null);
69
70         if (tarea != null) {
71             tareaRepository.delete(tarea);
72         }
73     }
74
75 }
```

El motivo de desarrollar hasta el punto descrito el backend es garantizar la escalabilidad de la aplicación, sobre todo con un enfoque de explotación de mayores cantidades de datos de valor mediante el diseño y cálculo automático de indicadores relacionados con características del usuario, las tareas que realiza, los proyectos en los que participa, etc. En otras palabras, el desarrollo del backend ha buscado plantear una estructura sólida sobre la que, posteriormente, poder implementar mejoras y ampliaciones de una manera sencilla, implementando los controladores necesarios y manteniendo los DTO como parte “pública” de la lógica de la base de datos.

El ejemplo de servicio de Tarea es bastante representativo porque los métodos **GetTareaById** y **save** se emplean tanto para recuperar tareas de la base de datos como para crear nuevas tareas y almacenarlas en ella. Finalmente, se puede observar en el controlador de tarea la implementación de los dos métodos que emplea la API:

- GET Tareas en base a parámetro usuario y parámetro fecha (Para cargar las tareas de cada usuario según la fecha concreta)
- POST nueva tarea

```

package com.ite.taskManager.controllers;

import java.time.LocalDate;

@RestController
public class TareaController {

    @Autowired
    private TareaService tareaService;

    @GetMapping("/tareas")
    public List<TareaDTO> getTareas(@RequestParam Long idUsuario, @RequestParam String fecha) {
        // Comprobacion recepcion conexion api //// 

        System.out.println("Recibida petición para obtener tareas");
        System.out.println("idUsuario: " + idUsuario);
        System.out.println("fecha: " + fecha);

        LocalDate localDate = LocalDate.parse(fecha);
        return tareaService.findByUsuario_IdAndFecha(idUsuario, localDate);
    }

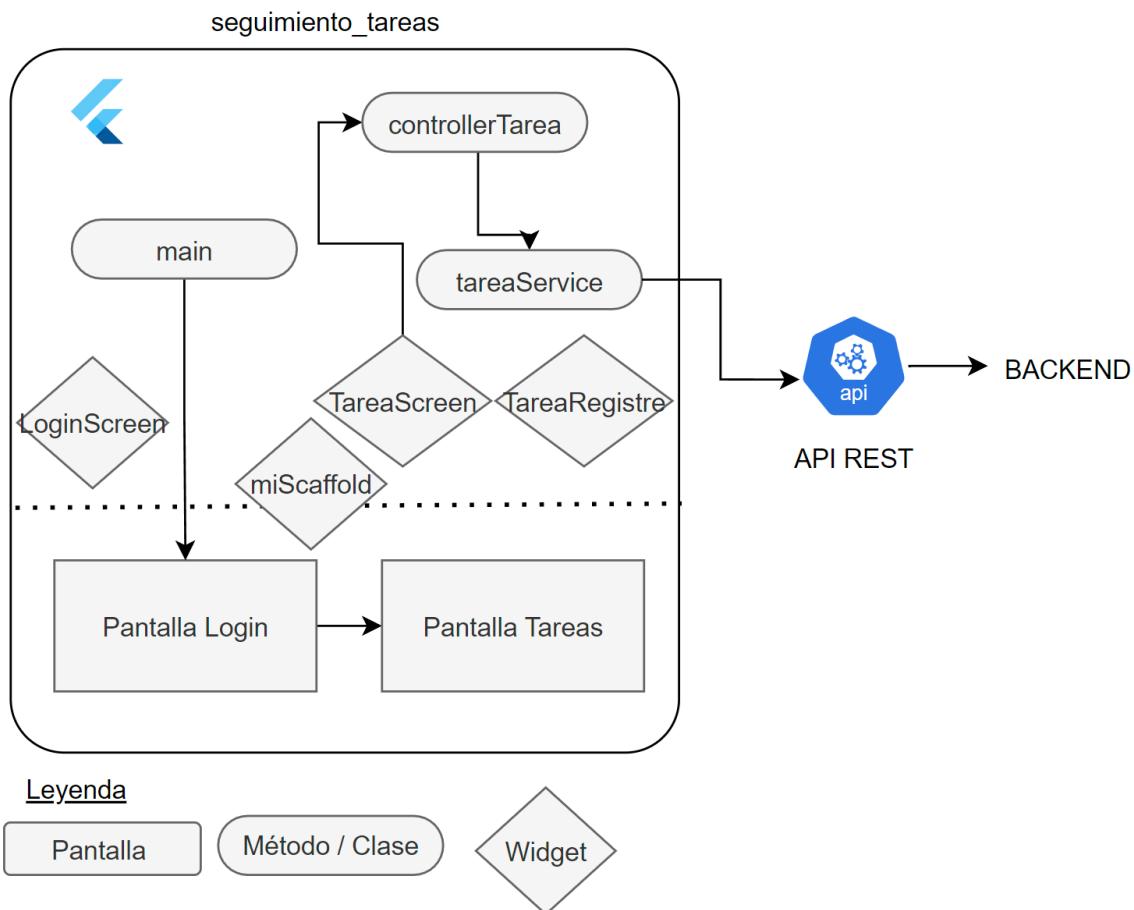
    @PostMapping("/tareas")
    public TareaDTO createTarea(@RequestBody TareaDTO tareaDTO) {
        System.out.println("Recibida petición para crear una tarea");
        return tareaService.save(tareaDTO);
    }
}

```

Desarrollo del Frontend

La interfaz de usuario de la aplicación se ha desarrollado en Flutter mediante Visual Studio Code y las extensiones pertinentes. Flutter, entre las ventajas que tiene, destaca el hecho de disponer de una potente librería de Widgets, de implementar la librería Material Design (Entre otras) y, sobre todo, de tener el soporte oficial y continuo de Google. La potencia de Flutter radica en que mediante un único desarrollo se dispone de una aplicación ejecutable tanto en Windows, como Mac, Android y navegador web, haciéndolo la herramienta multiplataforma perfecta. Como contraprestación, las implementaciones a veces pueden hacerse complejas y la curva de dificultad es, inicialmente, reseñable.

El proyecto se ha estructurado de manera similar (Aunque con matices) a un patrón MVC (Modelo Vista Controlador). A continuación, se muestra un esquema de los principales componentes, pantallas, widgets y métodos implementados:



Como se aprecia, la puerta de entrada a la aplicación es el main, que inmediatamente instancia a la pantalla LoginScreen(). Se ha configurado, además, para soporte en formato español, que para el caso afecta al idioma y formato de calendario, como se verá más adelante:

```
Run | Debug | Profile
void main() {
| runApp(const MyApp());
}

class MyApp extends StatelessWidget {
const MyApp({super.key});

// This widget is the root of your application.

@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'App seguimiento tareas',
localizationsDelegates: const [
GlobalMaterialLocalizations.delegate,
GlobalWidgetsLocalizations.delegate,
GlobalCupertinoLocalizations.delegate,
],
supportedLocales: const [
Locale('es', ''),
],
home: LoginScreen(),
); // MaterialApp
}
```

Otra implementación importante es mi_scaffold, que en este caso sirve como widget de navegación mediante una barra lateral común y omnipresente a todas las pantallas:

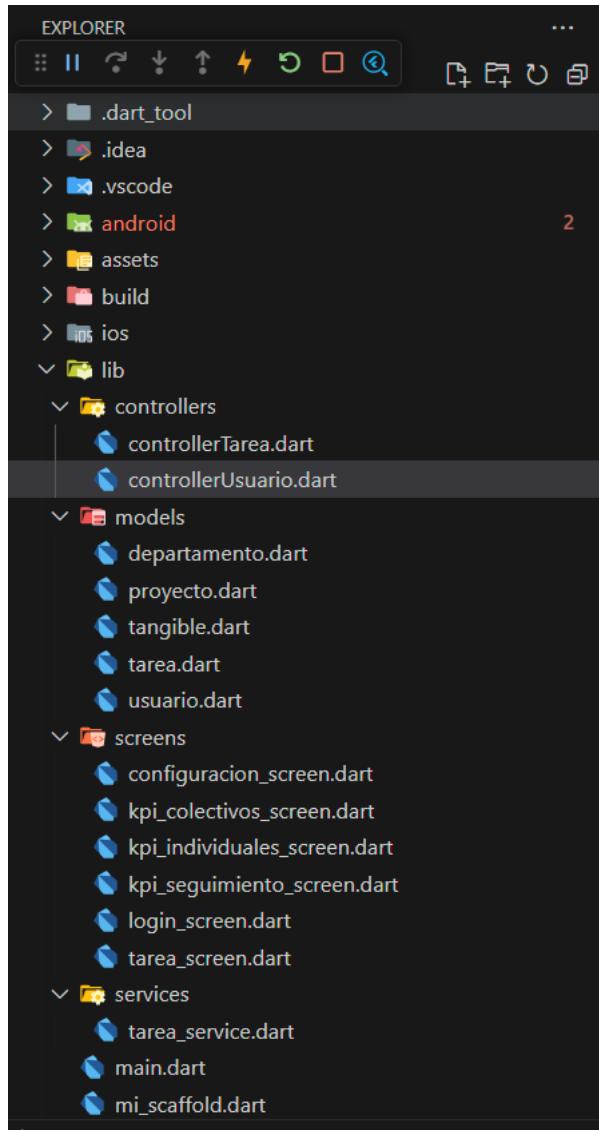
```
class MiScaffold extends StatelessWidget {
    final Widget body;
    final String title;
    final Widget? appBarLeading;
    final ControllerUsuario
        userController; // Controlar pantallas que se le muestran segun el usuario

    MiScaffold(
        {required this.body,
        required this.title,
        this.appBarLeading,
        required this.userController});

    @override
    Widget build(BuildContext context) {
        String? role = userController.getRole();
        String? username =
            userController.getUsername();
        return Scaffold(
            appBar: AppBar(
                title: Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    children: <Widget>[
                        appBarLeading ??
                            Container(), // Si appBarLeading es null, muestra un contenedor vacío
                        Text(title), // Titulo página
                        Text(username ?? 'Laura'), // Muestra el nombre del usuario
                        Image.asset('assets/images/logo.jpg',
                            height: 70.0), // Logo de la empresa // Image.asset
                    ],
                )));
}
```

De manera general, la aplicación se ha implementado con los colores corporativos y el logo de la empresa. La pantalla de login se ha implementado con un diseño atractivo y los elementos necesarios para gestionar el acceso autenticado de cada usuario. Por falta de tiempo en el desarrollo del proyecto por necesidad de acotar el alcance, la implementación exacta de la verificación de autenticación se deja pendiente de implementar en futuras actualizaciones.

En la siguiente captura se aprecia la estructura y las clases implementadas en la aplicación:



Los controladores se encargan de manejar la lógica de negocio de la aplicación, y serán llamados en cada uno de los widgets según proceda. El servicio de, en este caso, el objeto tarea, es el encargado de conectar con la aplicación backend en el lado servidor. Se han creado asimismo modelos de datos del resto de entidades (proyecto, tangible, etc), si bien se implementarán solo los atributos y la lógica que sea necesaria en cada uno de ellos. También se han creado varias pantallas que, en su momento, presentarán acceso identificado según el atributo “rol” del usuario.

A continuación, se muestra una captura de la pantalla de acceso autenticado:



Al pulsar “Entrar” se accede a la aplicación como tal, por defecto con el usuario ficticio “Laura” con id en base de datos id=2. No obstante, se ha previsto un controlador de Usuario que compruebe la identidad y rol de la persona que accede.

```
class ControllerUsuario {
    Usuario? usuarioActual;

    // Verificación credenciales usuario
    Future<bool> verifyPassword(String username, String password) async {
        // Solicitud a backend Springboot
        /*final response = await http.post(
            'http://backend/login' as Uri,
            body: jsonEncode(<String, String>{
                'username': username,
                'password': password,
            }),
        );

        if (response.statusCode == 200) {
            usuarioActual = Usuario(nombre: username, rol: 'normal');
            return true;
        } else {
            return false;
        }*/
        //////// Prueba manual usuarios //////
        usuarioActual = Usuario(nombre: 'Laura', rol: 'coordinadorArea');

        return true; //TODO
    }
}
```

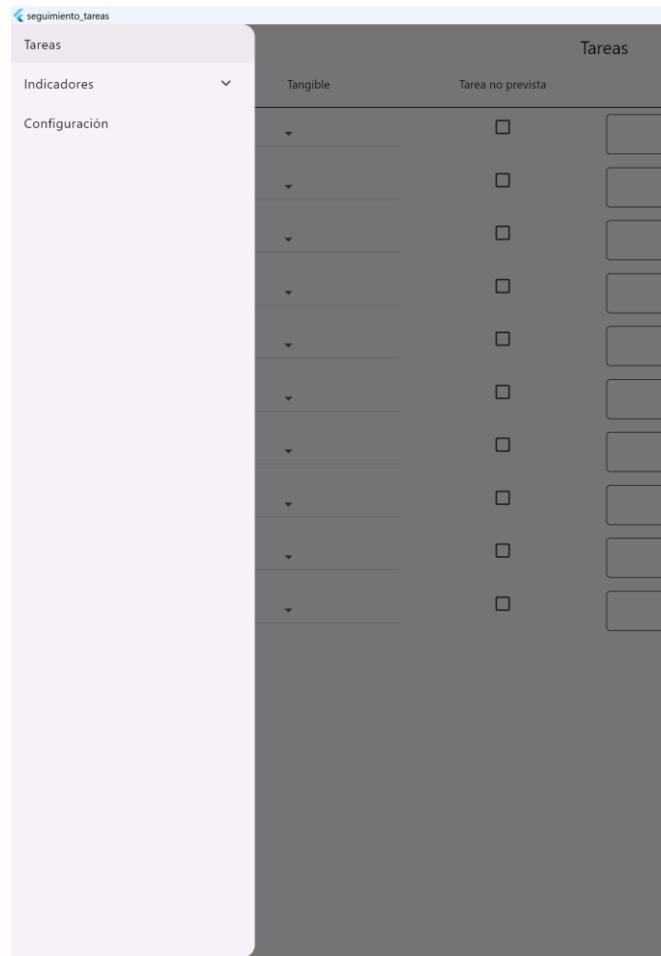
Como se ha explicado, se accede a la pantalla “Tareas”, que es la pantalla principal de la aplicación y el punto de intercambio de información principal del usuario:

Proyecto	Tangible	Tarea no prevista	Tarea	¿Reunión/evento?	Completada	Laura		Desviación
						Planificadas Total Planificadas: 0	Dedicadas Total Dedicadas: 0	
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
								

Esta pantalla se ha diseñado de la siguiente manera:

Proyecto	Tangible	Tarea no prevista	Tarea	¿Reunión/evento?	Completada	Laura		Desviación
						Planificadas Total Planificadas: 0	Dedicadas Total Dedicadas: 0	
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
▼	▼	<input type="checkbox"/>	[]	<input type="checkbox"/>	<input type="checkbox"/>	[]	[]	0
								

En el cuadro **azul** se representa el Scaffold, que es la interfaz común a todas las pantallas e incluye la barra de navegación lateral:



El cuadro **rojo** representa el widget TareaScreen

```
class TareaScreen extends StatefulWidget {
    @override
    _TareaScreenState createState() => _TareaScreenState();
}

class _TareaScreenState extends State<TareaScreen> {
    DateTime selectedDate = DateTime.now();

    TareaController tareaController = TareaController();
    ControllerUsuario userController = ControllerUsuario();

    late TareaRegistre tareaRegistre;

    @override
    void initState() {
        super.initState();
        tareaRegistre = TareaRegistre(
            controller: tareaController,
            actualizarEstadoWidgets: _actualizarEstadoWidgets);
        _cargarTareas(selectedDate);
    }
}
```

Dentro de dicho widget, además, se hace una instancia a TareaRegistre, que es el widget representado por el cuadro **naranja** y que se implementa como un widget especial en el que se crean una serie de métodos y widgets para mostrar un listado de tareas y sus características:

```

class TareaRegistre extends StatefulWidget {
    final TareaController controller;
    final VoidCallback actualizarEstadoWidgets;

    TareaRegistre(
        required this.controller, required this.actualizarEstadoWidgets);

    @override
    _TareaRegistreState createState() => _TareaRegistreState();
}

class _TareaRegistreState extends State<TareaRegistre> {
    @override
    void initState() {
        super.initState();
        for (var tarea in widget.controller.tareas) {
            tarea.planificadasController.addListener(() => setState(() {}));
            tarea.dedicadasController.addListener(() => setState(() {}));
        }
    }

    // CREACION WIDGETS //

    TableRow _crearFila(Tarea tarea) {
        return TableRow(
            children: [
                _crearDropdown(tarea.proyectoDescripcion,
                    ['Proyecto 1', 'Proyecto 2', 'Proyecto 3'], (newValue) {
                        setState(() {
                            tarea.proyectoDescripcion = newValue;
                        });
                    }),
                _crearDropdown(tarea.tangibleDescripcion,
                    ['Tangible 1', 'Tangible 2', 'Tangible 3'], (newValue) {
                        setState(() {
                            tarea.tangibleDescripcion = newValue;
                        });
                    }),
                _crearCheckbox(tarea.tareaNoPrevista, (newValue) {
                    setState(() {
                        tarea.tareaNoPrevista = newValue!;
                    });
                });
            ]
        );
    }
}

```

Partiendo del diseño creado en fases previas del proyecto, se han implementado las siguientes funciones de creación de Widgets:

- **crearFila**: crea una fila completa a partir de una tarea con todas sus características (Tangible, checkboxes, etc)
- **crearCheckbox**: dado que hay varias casillas de verificación, a partir de un valor booleano se creará (y recreará, según el caso) un checkbox
- Para los casos de Proyecto y Tangible no se ha implementado una conexión directa de base de datos, sino que se han preparado un widgets de tipo Dropdown para elegir de una preselección de proyectos a nivel indicativo, mediante la función **crearDropdown**
- Existen dos casillas que muestran texto:
 - o **crearTextFieldN** para crear una casilla de edición de texto que únicamente admita valores numéricos (para completar las horas planificadas y dedicadas en cada tarea)
 - o **crearTextFieldT** para los propios nombres de las tareas.

```

class _TareaRegistreState extends State<TareaRegistre> {
  TableRow _crearFila(Tarea tarea) {
    setState(() {
      tarea.reunionEvento = newValue!;
    });
  },
  _crearCheckbox(tarea.completada, (newValue) {
    setState(() {
      tarea.completada = newValue!;
    });
  }),
  _crearTextFieldN(tarea.planificadasController, (text) {}, null),
  _crearTextFieldN(tarea.dedicadasController, (text) {}, null),
  Padding(
    padding: const EdgeInsets.all(paddingValue),
    child: Text(
      '${tarea.desviacion}',
      textAlign: TextAlign.center,
    ), // Text
  ), // Padding
],
); // TableRow
}

Widget _crearDropdown(
  String? value, List<String> items, ValueChanged<String?> onChanged) {
  return Padding(
    padding: const EdgeInsets.all(paddingValue),
    child: DropdownButton<String>(
      value: items.contains(value)
        ? value
        : null, // Asegurar que si el valor no está en la lista, se establezca null
      items: items.map((String value) {
        return DropdownMenuItem<String>(
          value: value,
          child: Text(value),
        ); // DropdownMenuItem
      }).toList(),
      onChanged: onChanged,
    ), // DropdownButton
  ); // Padding
}

Widget _crearCheckbox(bool? value, ValueChanged<bool?> onChanged) {
  return Padding(
    padding: const EdgeInsets.all(paddingValue),
    child: Checkbox(
      value: value,
      onChanged: onChanged,
    ), // Checkbox
  ); // Padding
}

```

Los encabezados se han creado mediante un ListView con la primera línea de encabezados y una separación para el propio contenido de las tareas justo debajo. Se han implementado también dos contadores de cantidad de horas planificadas y dedicadas cada día.

```

@Override
Widget build(BuildContext context) {
  final totalPlanificadas = widget.controller.totalPlanificadas;
  final totalDedicadas = widget.controller.totalDedicadas;

  return Scaffold(
    body: ListView(
      children: [
        Table(
          border: const TableBorder(bottom: BorderSide()),
          children: [
            const TableRow(
              children: [
                Center(child: Text('Proyecto')),
                Center(child: Text('Tangible')),
                Center(child: Text('Tarea no prevista')),
                Center(child: Text('Tarea')),
                Center(child: Text('¿Reunión/evento?')),
                Center(child: Text('Completada')),
                Center(child: Text('Planificadas')),
                Center(child: Text('Dedicadas')),
                Center(child: Text('Desviación'))
              ],
            ), // TableRow
            TableRow(
              children: [
                Container(),
                Container(),
                Container(),
                Container(),
                Container(),
                Container(),
                Container(),
                Center(child: Text('Total Planificadas: $totalPlanificadas')),
                Center(child: Text('Total Dedicadas: $totalDedicadas')),
                Container(),
              ],
            ), // TableRow
          ],
        ), // Table
        ListView.builder(
          shrinkWrap: true,
          physics: const NeverScrollableScrollPhysics(),
          itemCount: widget.controller.tareas.length,
          itemBuilder: (context, index) {
            return Table(
              children: [_crearFila(widget.controller.tareas[index])],
            ); // Table
          }
        )
      ],
    ),
  );
}

```

Se ha implementado asimismo un botón en la parte inferior derecha que permite dar de alta nuevas tareas en la fecha actual. Es decir, el usuario introducirá una nueva tarea (Con los campos tangible y proyecto de manera indicativa) y esta tarea se dará de alta en la aplicación mediante los métodos de la API:



The screenshot shows a software interface for managing tasks. At the top, there's a header with the text "Tareas" and "Laura". Below the header is a table with columns: Proyecto, Tangible, Tarea no prevista, Tarea, (Reunión/Evento), Completada, Plazos Fijados, Total Pendientes, Dedicado, Total Dedicadas, and Devoción. There are six rows of data in the table. Below the table is a modal window titled "Agregar nueva tarea". It has fields for "Proyecto" (with a dropdown menu), "Tangible" (with a dropdown menu), and "Nombre de la tarea" (text input field). At the bottom of the modal are "Cancelar" and "Guardar" buttons. A small green circular icon is located in the bottom right corner of the main window area.

El controlador de tarea lleva a cabo varias funciones, todas ellas asíncronas por la necesidad de interactuar con el backend de la aplicación (Y, en última instancia, con la base de datos).

```

class TareaController {
    // Inicio servicio

    final TareaService _tareaService = TareaService();

    // Generacion listado de tareas vacío (Por defecto 10)

    List<Tarea> tareas = List.generate(10, (i) => Tarea());

    Future<void> cargarTareas(DateTime fecha) async {
        List<Tarea> tareasCargadas = await _tareaService.fetchTareasPorFecha(fecha);

        // Rellenar la lista tareas con las tareas obtenidas, manteniendo 10 elementos
        for (int i = 0; i < tareas.length; i++) {
            if (i < tareasCargadas.length) {
                tareas[i] = tareasCargadas[i];
            } else {
                tareas[i] = Tarea();
            }
        }
    }

    Future<void> crearTarea(Tarea tarea) async {
        try {
            await _tareaService.crearTarea(tarea);
            // Actualizar la lista de tareas después de crear una nueva tarea
            await cargarTareas(DateTime.now());
        } catch (e) {
            print('Error al crear la tarea: $e');
        }
    }

    void updateTarea(int index, Tarea tarea) {
        tareas[index] = tarea;
    }

    // Formateadores de entrada para asegurar que solo se ingresen números
    final planificadasFormatter = FilteringTextInputFormatter.digitsOnly;
    final dedicadasFormatter = FilteringTextInputFormatter.digitsOnly;

    // Funcion de color según si completada o no
    Color getColor(Tarea tarea) {
        return tarea.completada ? Colors.grey : Colors.white;
    }
}

```

La función **cargarTareas**, que llama al servicio para conectar con la API mediante el método `fetchTareasPorFecha` (y dada la fecha de consulta que el usuario proporciona a través del selector del calendario)

La función **crearTarea**, que se invoca cuando se crea una nueva tarea (llamando a su vez, cuando corresponde, a `cargarTarea`), registrando esta tarea en la base de datos mediante un POST.

La integración final con el backend se realiza mediante la clase TareaService, en la que se implementa la conexión HTTP mediante la librería correspondiente y la configuración de URL, puerto y configuración de los endpoints realizada en el lado backend:

```

class TareaService {
    final String baseUrl = 'http://127.0.0.1:9877'; // URL, modificar si cambia

    Future<List<Tarea>> fetchTareasPorFecha(DateTime fecha) async {
        // Formatea la fecha a una cadena de texto en el formato 'yyyy-MM-dd'
        final String fechaFormatada = DateFormat('yyyy-MM-dd').format(fecha);
        final response = await http
            .get(Uri.parse('$baseUrl/tareas?idUsuario=2&fecha=$fechaFormatada'));

        // Imprime la respuesta del servidor en la consola
        print('Response status: ${response.statusCode}');
        print('Response body: ${response.body}');

        if (response.statusCode == 200) {
            List<dynamic> body = jsonDecode(response.body);
            List<Tarea> tareas =
                body.map((dynamic item) => Tarea.fromJson(item)).toList();
            return tareas;
        } else {
            throw Exception('No se han podido cargar las tareas');
        }
    }

    // Crear tarea nueva
    Future<void> crearTarea(Tarea tarea) async {
        final response = await http.post(
            Uri.parse('$baseUrl/tareas'),
            headers: <String, String>{
                'Content-Type': 'application/json; charset=UTF-8',
            },
            body: jsonEncode(tarea.toJson()),
        );

        if (response.statusCode != 200) {
            throw Exception('No se ha podido crear la tarea');
        }
    }
}

```

Como se aprecia, se decodifica el archivo JSON recibido de las peticiones mediante una implementación en el modelo de datos de Tarea, que tiene la siguiente estructura:

```

double planificadas = 0;
double dedicadas = 0;
//int desviacion = 0;

// Controladores de texto para los campos Planificadas y De
final planificadasController = TextEditingController();
final dedicadasController = TextEditingController();
final nombreTareaController = TextEditingController();

// Constructor
Tarea([
    this.proyectoDescripcion,
    this.tangibleDescripcion,
    this.tareaNoPrevista = false,
    this.nombreTarea = '',
    this.reunionEvento = false,
    this.completada = false,
    this.planificadas = 0,
    this.dedicadas = 0,
])
{
    nombreTareaController.text =
        nombreTarea; // Inicializa el controlador con el valo
}

// Constructor para crear una Tarea desde un JSON
factory Tarea.fromJson(Map<String, dynamic> json) {
    return Tarea(
        proyectoDescripcion: json['proyectoDescripcion'],
        tangibleDescripcion: json['tangibleDescripcion'],
        tareaNoPrevista: json['tareaNoPrevista'] ?? false,
        nombreTarea: json['descripcion'],
        reunionEvento: json['reunion'] ?? false,
        completada: json['completada'] ?? false,
        planificadas: json['horasPrevistas'],
        dedicadas: json['horasEjecutadas'] ?? 0,
    );
}

// Método para convertir una Tarea en JSON
Map<String, dynamic> toJson() {
    return {
        'proyectoDescripcion': proyectoDescripcion,
        'tangibleDescripcion': tangibleDescripcion,
        'tareaNoPrevista': tareaNoPrevista,
        'descripcion': nombreTarea,
        'reunion': reunionEvento,
        'completada': completada,
        'horasPrevistas': planificadas,
        'horasEjecutadas': dedicadas,
    };
}

```

Como se aprecia, se han implementado tanto los constructores como los métodos necesarios para enviar y recibir la información relativa a las tareas en archivos JSON. Se muestra a modo de ejemplo un archivo correspondiente a una petición de varias tareas y sus características asignadas a un día concreto:

```
flutter: Response status: 200
flutter: Response body: [{"idTarea":6,"descripcion":"Llamada con el cliente","fecha":"2024-05-23","horasPrevistas":1.0,"horasEjecutadas":null,"completada":false,"reunion":false,"usuarioId":2,"tangibleId":2,"tangibleDescripcion":"Material de publicidad","proyectoDescripcion":null},{"idTarea":7,"descripcion":"Estudio casos de uso","fecha":"2024-05-23","horasPrevistas":1.0,"horasEjecutadas":null,"completada":false,"reunion":true,"usuarioId":2,"tangibleId":2,"tangibleDescripcion":"Material de publicidad","proyectoDescripcion":null},{"idTarea":8,"descripcion":"Definir accion","fecha":"2024-05-23","horasPrevistas":1.0,"horasEjecutadas":null,"completada":true,"reunion":true,"usuarioId":2,"tangibleId":2,"tangibleDescripcion":"Material de publicidad","proyectoDescripcion":null},{"idTarea":9,"descripcion":"Trabajar en el informe de marketing","fecha":"2024-05-23","horasPrevistas":4.0,"horasEjecutadas":null,"completada":false,"reunion":true,"usuarioId":2,"tangibleId":2,"tangibleDescripcion":"Material de publicidad","proyectoDescripcion":null}]
```

5.4. Evaluación del sistema

Para la evaluación, se llevarán a cabo las siguientes pruebas:

- Pruebas de visualización de valores en la base de datos, precargando tareas y usuarios ficticios
- Pruebas de petición al backend, específicamente a los endpoints programados para comprobar su funcionalidad, para lo que se empleará Postman
- Pruebas de navegación y respuesta de la interfaz de usuario
- Pruebas de integración: recuperación de tareas ya existentes en base de datos y creación de nuevas tareas, desde el Frontend hasta la base de datos y viceversa

Se detallan capturas de los resultados de cada una de las pruebas realizadas:

Pruebas de visualización de valores en la base de datos

Se han generado una serie de tareas ficticias:

```
-- Insertar tareas
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Configuración del servidor', '2023-05-01', 8.0, 7.5, TRUE, FALSE, 2, 1);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Desarrollo del modulo de autenticación', '2023-05-02', 16.0, 15.0, TRUE, FALSE, 3, 1);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Diseño de flyers', '2023-05-03', 10.0, NULL, FALSE, FALSE, 4, 2);
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Reunión de coordinación', '2023-05-04', 2.0, 2.0, TRUE, TRUE, 1, 2);

INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Poner al dia correos', '2024-05-23', 2.0, NULL, FALSE, FALSE, 1, 1);

INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Llamada con el cliente', '2024-05-23', 1.0, NULL, FALSE, FALSE, 2, 2);

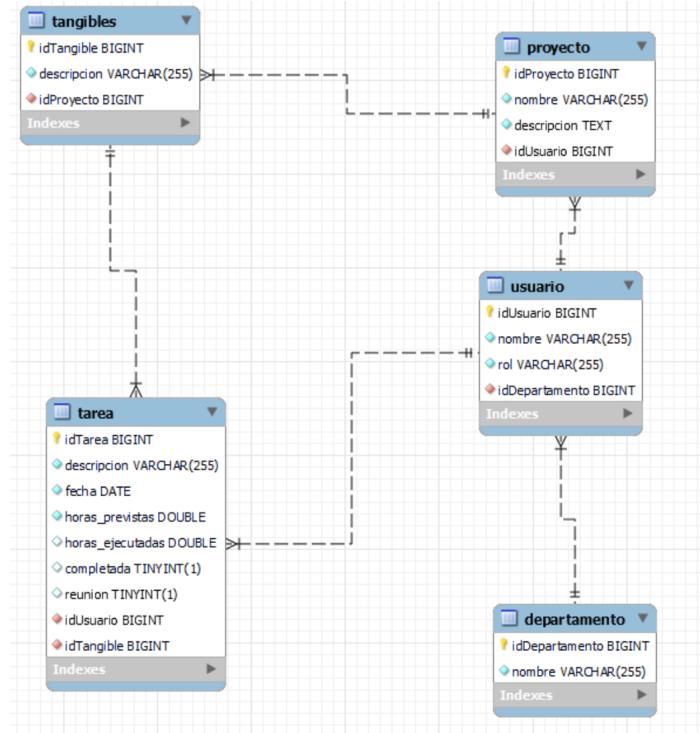
INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Estudio casos de uso', '2024-05-23', 1.0, NULL, FALSE, TRUE, 2, 2);

INSERT INTO tarea (descripcion, fecha, horas_previstas, horas_ejecutadas, completada, reunion, idUsuario, idTangible)
VALUES ('Definir acción', '2024-05-23', 1.0, NULL, TRUE, TRUE, 2, 2);
```

Mediante MySQL Workbench se comprueba que la estructura de la base de datos es correcta y las tareas se rellenan como debe ser:

	idTarea	descripcion	fecha	horas_previstas	horas_ejecutadas	completada	reunion	idUser	idTangible
▶	1	Configuración del servidor	2023-05-01	8	7.5	1	0	2	1
	2	Desarrollo del modulo de autenticación	2023-05-02	16	15	1	0	3	1
	3	Diseño de flyers	2023-05-03	10	NULL	0	0	4	2
	4	Reunión de coordinación	2023-05-04	2	2	1	1	1	2
	5	Poner al dia correos	2024-05-23	2	NULL	0	0	1	1
	6	Llamada con el cliente	2024-05-23	1	NULL	0	0	2	2
	7	Estudio casos de uso	2024-05-23	1	NULL	0	1	2	2
	8	Definir acción	2024-05-23	1	NULL	1	1	2	2
	9	Trabajar en el informe de marketing	2024-05-23	4	NULL	0	1	2	2
	10	Revisar reuniones	2024-05-29	3	NULL	0	0	2	2
	11	Revisar propuestas	2024-05-28	3	NULL	0	0	2	2
	13	Imprimir documentos	2024-05-29	0	0	0	0	2	1
	14	Comprar ordenador	2024-05-29	0	0	0	0	2	1
	15	Estado del arte	2024-05-30	0	0	0	0	2	1
*	HULL	NULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

El esquema de la base de datos tal y como se había presentado antes confirma también que la estructura es correcta:



Pruebas de petición al backend

Se realizan peticiones a los dos endpoints configurados en el controlador de tareas:

```
@RestController
public class TareaController {

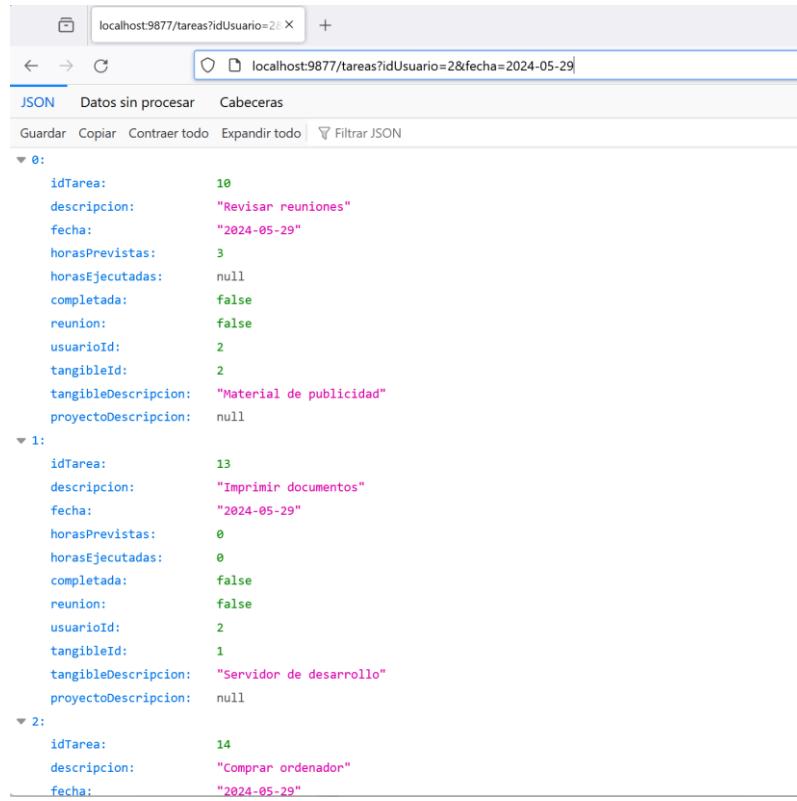
    @Autowired
    private TareaService tareaService;

    @GetMapping("/tareas")
    public List<TareaDTO> getTareas(@RequestParam Long idUsuario, @RequestParam String fecha) {
        /// Comprobacion recepcion conexion api /////
        System.out.println("Recibida petición para obtener tareas");
        System.out.println("idUsuario: " + idUsuario);
        System.out.println("fecha: " + fecha);

        LocalDate localDate = LocalDate.parse(fecha);
        return tareaService.findByUsuario_IdAndFecha(idUsuario, localDate);
    }

    @PostMapping("/tareas")
    public TareaDTO createTarea(@RequestBody TareaDTO tareaDTO) {
        System.out.println("Recibida petición para crear una tarea");
        return tareaService.save(tareaDTO);
    }
}
```

La petición GET se comprueba desde el propio navegador, con el usuario con id=2 y para, por ejemplo, el 29/05:



```

[{"idTarea": 10, "descripcion": "Revisar reuniones", "fecha": "2024-05-29", "horasPrevistas": 3, "horasEjecutadas": null, "completada": false, "reunion": false, "usuarioId": 2, "tangibleId": 2, "tangibleDescripcion": "Material de publicidad", "proyectoDescripcion": null}, {"idTarea": 13, "descripcion": "Imprimir documentos", "fecha": "2024-05-29", "horasPrevistas": 0, "horasEjecutadas": 0, "completada": false, "reunion": false, "usuarioId": 2, "tangibleId": 1, "tangibleDescripcion": "Servidor de desarrollo", "proyectoDescripcion": null}, {"idTarea": 14, "descripcion": "Comprar ordenador", "fecha": "2024-05-29", "horasPrevistas": null, "horasEjecutadas": null, "completada": null, "reunion": null, "usuarioId": null, "tangibleId": null, "tangibleDescripcion": null, "proyectoDescripcion": null}]
  
```

Para probar el POST si que será necesario, dado que recibe un TareaDTO, realizarlo desde el propio Frontend.

Pruebas de navegación y respuesta de la interfaz de usuario

Estas pruebas simplemente se han testeado de manera manual empleando la interfaz y de manera iterativa con cada desarrollo. Se adjuntan algunas capturas a modo de evidencia:

The screenshot shows a Windows application window titled 'seguimiento_tareas'. The main content area features a large yellow circular logo on the left containing a stylized figure, and to its right, the letters 'ITE' in a large, bold, dark gray font. Below 'ITE', the text 'INSTITUTO TECNOLÓGICO DE LA ENERGÍA' is written in a smaller, dark gray font. At the bottom of the window, there are two input fields: one for 'Usuario' (User) with a person icon and another for 'Contraseña' (Password) with a lock icon. A purple 'Entrar' (Enter) button is positioned between the fields. A small red ribbon in the top right corner of the window says 'DEBUG'.

The screenshot shows a digital interface for managing tasks. On the left, there is a grid of 12 task cards arranged in four rows and three columns. Each card has a checkbox in the top-left corner and two empty rectangular fields for notes. The first five cards in each row have a value of 0 to their right. The last card in each row has a green button labeled '+ Agregar nueva tarea para hoy' (Add new task for today) with a small upward arrow icon. A large green '+' button is located at the bottom right of the grid. On the right side of the screen, a dark gray modal window titled 'Agregar nueva tarea' (Add new task) is open. It contains fields for 'Proyecto' (Project), 'Tangible' (Tangible), and 'Nombre de la tarea' (Task name). The modal has a white background and a close button in the top-right corner.

The screenshot shows a task management interface with a modal dialog for selecting a date. The main view has columns for 'Proyecto' (Project), 'Tangible' (Tangible), 'Tarea no prevista' (Unplanned Task), 'Tarea' (Task), '¿Reunión/evento?' (Meeting/Event), 'Completada' (Completed), and 'Planificado' (Planned). A specific row is highlighted with a red box, and a date selection dialog is overlaid on it. The dialog title is 'Seleccionar fecha' (Select Date) and shows 'mayo de 2024' (May 2024). The calendar grid includes days from 1 to 31, with '31' circled in blue. Buttons at the bottom are 'Cancelar' (Cancel) and 'ACEPTAR' (Accept).

This screenshot shows a list of tangible tasks under the 'Tareas' (Tasks) section. The header includes a date filter '2024-05-30'. The list contains three items: 'Tangible 1', 'Tangible 2', and 'Tangible 3', each with a checkbox column.

This screenshot shows a list of individual indicators under the 'Indicadores' (Indicators) section. The header includes a date filter '2024-05-30'. The list contains two items: 'Individuales' and 'Configuración', each with a checkbox column.

Tareas

Tarea no prevista	Tarea	¿Reunión/evento?	Completada
<input type="checkbox"/>	Estado del artr	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>

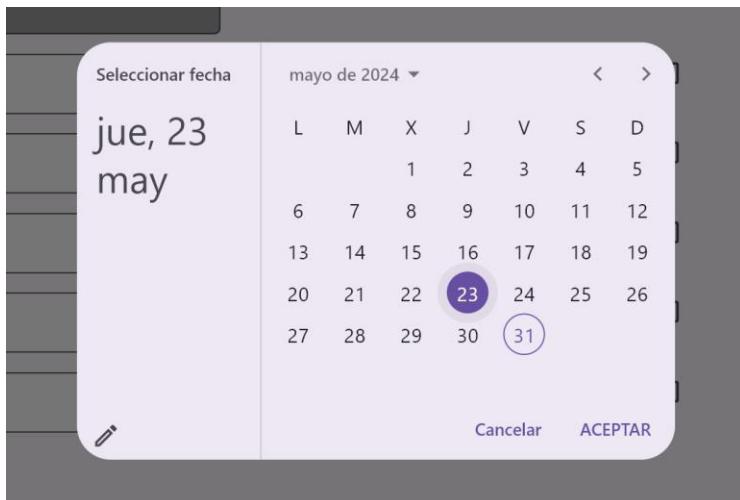
Laura



Completada	Planificadas Total Planificadas: 8	Dedicadas Total Dedicadas: 6	Desviación
<input type="checkbox"/>	5	2	-3
<input checked="" type="checkbox"/>	3	4	1
<input checked="" type="checkbox"/>			0
-			0

Pruebas de integración

Se prueba seleccionando el día 23 para comprobar si cargan las tareas:



Se comprueba que, en efecto, cargan correctamente, con la única limitación que hay que pulsar en algún widget de la pantalla central:

Proyecto	Tangible	Tarea no prevista	Tareas	¿Reunión/evento?	Completada	Laura
▼	▼	<input type="checkbox"/>	Llamada con el cliente	<input type="checkbox"/>	<input type="checkbox"/>	
▼	▼	<input type="checkbox"/>	Estudio casos de uso	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
▼	▼	<input type="checkbox"/>	Definir acción	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
▼	▼	<input type="checkbox"/>	Trabajar en el informe de marketing	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
▼	▼	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
▼	▼	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	
▼	▼	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	

Comprobamos que, en efecto, en base de datos coinciden las tareas para ese día con el usuario Id=2

idTarea	descripcion	fecha	horas_previstas	horas_ejecutadas	completada	reunion	idUsuario	idTangible
1	Configuración del servidor	2023-05-01	8	7.5	1	0	2	1
2	Desarrollo del modulo de autenticación	2023-05-02	16	15	1	0	3	1
3	Diseño de flyers	2023-05-03	10	NULL	0	0	4	2
4	Reunión de coordinación	2023-05-04	2	2	1	1	1	2
5	Poner al dia correos	2024-05-23	2	NULL	0	0	1	1
6	→ Llamada con el cliente	2024-05-23	1	NULL	0	0	2 ↙	2
7	→ Estudio casos de uso	2024-05-23	1	NULL	0	1	2 ↙	2
8	→ Definir accion	2024-05-23	1	NULL	1	1	2 ↙	2
9	→ Trabajar en el informe de marketing	2024-05-23	4	NULL	0	1	2 ↙	2

Por último, se prueba a dar de alta una nueva tarea:

Agregar nueva tarea

Proyecto
Alpha

Tangible
Beta

Nombre de la tarea
Escribir carta solicitud

Vemos que la propia tarea ya aparece listada, lo que debería indicarnos que es correcto:

seguimiento_tareas				
≡	📅 2024-05-31	Tareas		
Proyecto	Tangible	Tarea no prevista	Tarea	¿Reunión/ev?
▼	▼	□	Escribir carta soli	□
▼	▼	□		□
		□		□

Aun así, se comprueba que, en efecto, se ha creado correctamente en base de datos:

Result Grid		Edit		Export/Import		Wrap Cell Content				
		idTarea	descripcion	fecha	horas_previstas	horas_ejecutadas	completada	reunion	idUser	idTangible
▶	1	Configuración del servidor		2023-05-01	8	7.5	1	0	2	1
	2	Desarrollo del modulo de autenticación		2023-05-02	16	15	1	0	3	1
	3	Diseño de flyers		2023-05-03	10	NULL	0	0	4	2
	4	Reunión de coordinación		2023-05-04	2	2	1	1	1	2
	5	Poner al dia correos		2024-05-23	2	NULL	0	0	1	1
	6	Llamada con el cliente		2024-05-23	1	NULL	0	0	2	2
	7	Estudio casos de uso		2024-05-23	1	NULL	0	1	2	2
	8	Definir accion		2024-05-23	1	NULL	1	1	2	2
	9	Trabajar en el informe de marketing		2024-05-23	4	NULL	0	1	2	2
	10	Revisar reuniones		2024-05-29	3	NULL	0	0	2	2
	11	Revisar propuestas		2024-05-28	3	NULL	0	0	2	2
	13	Imprimir documentos		2024-05-29	0	0	0	0	2	1
	14	Comprar ordenador		2024-05-29	0	0	0	0	2	1
	15	Estado del artr		2024-05-30	0	0	0	0	2	1
*	16	Escribir carta solicitud		2024-05-31	0	0	0	0	2	1
		NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Con esto se dan por concluidas las pruebas de manera satisfactoria.

6. Implantación de la solución

6.1. Fases de implantación

La implantación de la solución se realizará de manera escalonada a los diferentes departamentos de la empresa. A pesar de que se han modelado los diferentes departamentos y la posibilidad de dar de alta diferentes responsables de proyecto y responsables de área, se comenzará por el área de Eficiencia en los Usos de la Energía, por diferentes razones:

- Es el área que actualmente emplea el método organizativo del Excel de tareas. La transición para pasar a emplear esta aplicación es inmediata y los usuarios ya están fidelizados
- El alumno pertenece a dicha área, lo que facilita estar cerca y formar parte del grupo piloto
- El área reúne una cantidad importante de personas y los diferentes roles para poder implantar y adoptar la solución

Se plantea emplear la aplicación durante los primeros meses para generar conclusiones acerca de su uso, recoger la opinión de los usuarios y corregir bugs y realizar posibles mejoras.

La aplicación deberá, además, ser alojada en los servidores de la empresa, de manera similar a otras implementaciones ya existentes. Un ejemplo claro es un portal de incidencias desarrollado en la propia organización para gestionar las peticiones de incidencias que se aloja en dichos servidores, de manera que es necesario estar conectado a la intranet de la empresa para poder emplearlo.

6.2. Formación, comunicación y soporte

Se desarrollará un **manual de usuario simplificado** explicando el uso y las funcionalidades de la misma, con capturas e indicaciones. Además, se dispondrá una **dirección de correo** de consulta (en este caso a cargo del propio alumno) para la resolución de dudas más específicas y soporte de incidencias.

Se realizarán **encuestas de satisfacción** tras los primeros meses de uso para recabar el feedback de los usuarios

6.3. Mejoras y futuras implementaciones

Se han planificado y considerado las siguientes mejoras e implementaciones, a realizar en futuras versiones de la aplicación:

- Implementación del control de acceso autenticado y cifrado de credenciales tanto desde Backend como desde Frontend
- Implementación de funcionalidades básicas en pantalla de configuración según el perfil de usuario. Para el rol de coordinador de proyecto y coordinador de área, capacidad de generar nuevos tangibles y proyectos y eliminar (Previo diálogo de confirmación) los existentes
- Mejoras de carga de las tareas y proceso de dar de alta nuevas tareas. Vinculación efectiva de tareas con tangibles y proyectos dados de alta. Mejora de la navegación y usabilidad de la aplicación
- Implementación de KPIs y visualización por pantalla. Dependiente de detalle de las funcionalidades por usuarios finales
- Como mejora de importancia a la hora de gestionar los estados de los Widgets de la app Flutter, convendría adaptar el diseño general de la aplicación al patrón Provider, que en este sentido facilita mucho la gestión y actualización automática de dichos estados.

7. Conclusiones

Como conclusiones, en el proyecto se ha buscado desde su concepción realizar todo el ciclo de desarrollo de una aplicación, desde la fase de diagnóstico de necesidad, realización de especificaciones y diseño básico, hasta la implementación y puesta en marcha de todos los eslabones de la cadena (Lógica de datos y acceso a los mismos, desde el backend, y visualización de la información e interacción con el usuario final, desde el frontend). Como conclusión, a pesar de que el alcance no ha sido el inicialmente previsto, el alumno ha podido entender la importancia de todos los aspectos del desarrollo, las complejidades imprevistas e inherentes al proceso de implementación de cualquier funcionalidad, así como el coste global de desarrollar una aplicación funcional, segura y, sobre todo, acorde a los requisitos, funcionalidades y necesidades finales del usuario.

En este sentido, el proyecto ha proporcionado al alumno, no solo una consolidación del conocimientos sobre desarrollo software, sino también una visión integral, holística y representativa sobre la complejidad e importancia del trabajo de desarrollo de software y la importancia de la especialización para maximizar la productividad. Otra de las principales conclusiones que el alumno extrae del trabajo es la importancia de hacer una buena identificación de requisitos del cliente (Que en muchas ocasiones puede no acabar incluso definiendo bien sus necesidades), ya que el proceso de desarrollo tiene sus particularidades y se hace especialmente complejo corregir desarrollos ya en curso en caso de errar en dichas especificaciones.

8. Bibliografía

- <https://refactoring.guru/es/design-patterns/facade>
- https://www.tutorialspoint.com/spring/spring_quick_guide.htm
- <https://goalkicker.com/SpringFrameworkBook/SpringFrameworkNotesForProfessionals.pdf>
- https://aulas.edu.gva.es/semipresencial/pluginfile.php/2176800/mod_resource/content/2/6.2.Provider.html
- https://aulas.edu.gva.es/semipresencial/pluginfile.php/2171221/mod_resource/content/1/5.1.FutureBuilder.html
- https://aulas.edu.gva.es/semipresencial/pluginfile.php/2149321/mod_resource/content/2/3.3.CicleVida.html
- https://aulas.edu.gva.es/semipresencial/pluginfile.php/2149321/mod_resource/content/2/3.7.Temes.html
- https://aulas.edu.gva.es/semipresencial/pluginfile.php/1706265/mod_resource/content/0/HibernateNotesForProfessionals.pdf
- https://aulas.edu.gva.es/semipresencial/pluginfile.php/1706282/mod_resource/content/0/Unit%205%20-%20Spring.pdf
- <https://docs.flutter.dev/>