Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Aplicación para axudar na xestión e monitorización das vendimias

Estudante: Mateo Tilves Freijeiro

Dirección: Juan Raposo Santiago

A Coruña, agosto de 2024.

Agradecimientos

Resumen

El trabajo consiste en el diseño e implementación de una aplicación para ayudar en la gestión de procesos de negocio de una empresa viticultora, en específico en el proceso de la recogida de la uva (vendimia).

Durante el proceso de vendimia la plantilla de empleados crece enormemente por lo que una aplicación para la ayuda de la gestión de los trabajos de los empleados puede ayudar mucho en el control y trazabilidad de la uva y del trabajo realizado.

Habrá tres tipos de roles; administrador, capataz y tractoristas. Un mismo usuario podrá ejercer rol de capataz y tractorista. Se requerirá autenticación de estos usuarios.

La aplicación tendrá un catálogo de zonas de recogida y líneas de parras asociadas a cada zona, y estas líneas de parras con datos sobre el tipo de uva, el tipo de formación de la línea, edad, metros de línea, etc. Esta información podrá ser creada y modificada por empleados de la empresa con rol de administrador.Para la gestión del trabajo de la vendimia, los capataces podrán, mediante el uso de la aplicación, informar del trabajo realizado en las distintas líneas de parras, tanto de tareas de mantenimiento previa a la vendimia, como de la recogida de la uva.

Los tractoristas tendrán un rol especial, ya que serán notificados una vez se trabaje por completo una cantidad determinada de líneas de parras, con información detallada de la zona y líneas que deben recoger.

Para la facilidad de uso de la aplicación por parte de los capataces, estos podrán registrar el trabajo de cada línea de parras con un código QR.

Los capataces podrán utilizar la aplicación para añadir trabajos asociados a cada una de las líneas de parras, asignando recursos (personal de vendimia), tipo de trabajo, y opcionalmente añadir comentarios sobre el trabajo realizado en una línea de parras. Para facilitar el uso se puede leer un código QR para iniciar este proceso, aunque también tendrá la posibilidad de buscar la zona y línea manualmente.

La aplicación constará de un backend y un frontend. El backend estará implementado en Java, y será accesible mediante una API REST basada en Spring y desarrollada con una metodología API First con el uso de la herramienta OpenAPI.

Abstract

The work consists in the design and implementation of an application to help in the management of business processes of a wine-growing company, specifically in the process of grape harvesting (grape harvest).

During the harvesting process, the number of employees grows enormously, so an application for the application to help manage the work of the employees can be of great help in the control and traceability of the grapes. control and traceability of the grapes and the work carried out.

The application shall have a catalogue of harvesting zones and vine lines associated to each zone, and these vine lines with data on the type of grape, the type of formation used on the line, age, metres of line, etc. This information can be created and modified by company employees with the role of administrator. For the management of harvest work, foremen will be able, by using the application, to report on the work carried out on the different lines, report on the work carried out on the different lines of vines, both in terms of maintenance tasks prior to the harvest, as well as tasks prior to the harvest, as well as grape harvesting.

Tractor drivers will have a special role, as they will be notified once a certain number of vine lines have been completely worked. number of lines of vines are completely worked, with detailed information on the area and lines where to pick up the boxes loaded with grapes.

For the ease of use of the application by the foremen, they will be able to register the work of each line of vines with a QR code, which will be on each line of vines.

Foremen will be able to use the application to add jobs associated with each of the lines of vines, assigning resources (harvesting personnel, type of work, and optionally adding resources (harvesting personnel), type of work, and optionally add comments on the work carried out on a line of vines. comments on the work carried out on a line of vines. For ease of use, a QR code can be read to start this process, although you can also search for the area and line manually.

There will be three types of roles; administrator, foreman and tractor drivers. The same user will be able to play the role of foreman and tractor driver. Authentication of these users will be required. The application will consist of a backend and a frontend. The backend will be implemented in Java, and it will be accessible via a REST API based on Spring and developed with an API First methodology using the OpenAPI tool.

Palabras clave:

- JPA
- Hibernate
- Spring
- Flutter
- OpenAPI
- OAS
- OpenAPI Generator
- Scrumban
- Funcionalidad
- Vendimia
- Administrador
- Capataz
- Tractorista

Keywords:

- JPA
- Hibernate
- Spring
- Flutter
- OpenAPI
- OAS
- OpenAPI Generator
- Scrumban
- Feature
- Harvest
- Administrator
- Foreman
- Tractor driver

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

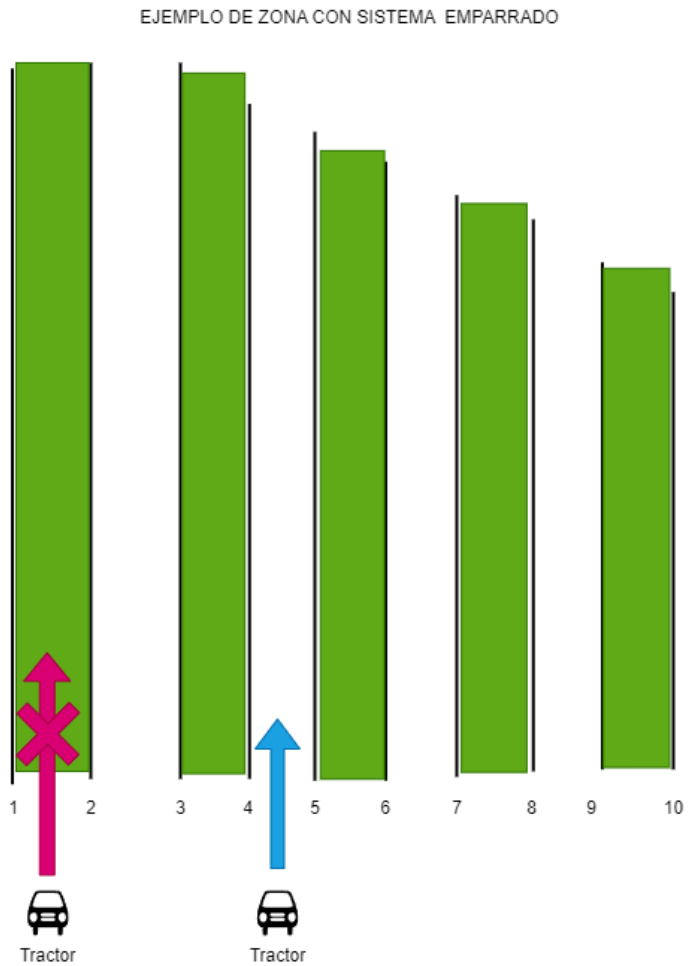
La vendimia es la recolección de la uva, que se procesa y se trata para obtener vino. Esta actividad milenaria se realiza normalmente en los meses de septiembre y octubre, cuando se recogen los frutos maduros de la vid. Es un evento en el que se reúne mucha gente para ”apañar” la uva, y motivo de festividad [1] en muchos puntos de Galicia (y del mundo). Normalmente, en viñas domésticas no muy extensas, el trabajo es muy manual y no requiere de mucha gente y tiempo para recolectar toda la uva. Lo mas habitual es que hagan uso de un sistema de emparrado de conducción horizontal, típico en el rural gallego, especialmente en las Rías Baixas, donde las plantas de vids se forman con una estructura de pasillo de ”arcos” 1.1a. A pesar de su belleza en el rural, normalmente están construidos para la ”comodidad” de cada dueño de la vid, por lo que puede ser difícil encontrar dos arcos a la misma altura. Esto hace que la recolección de la uva sea mas difícil para personas de distinta altura y además dificulta el uso de maquinaria especifica como los tractores 1.1b.

Las empresas viticultoras con viñas extensas recolectan muchas toneladas de uva, dependiendo del tamaño del terreno utilizado para el cultivo de las vides, el tipo de uva y su rendimiento [2]. A pesar de que muchas empresas siguen teniendo vides en formación de emparrado horizontal, aprovechando así vides antiguas para la producción de vino de mayor calidad, se suele utilizar una formación de parras verticales, denominada formación espaldera 1.2a. Esta formación en espaldera es mucho mas eficiente para trabajar, tanto para operarios de mantenimiento en tareas como podas, control de plagas y enfermedades, y el uso de maquinaria para tareas fitosanitarias, como para operarios de la vendimia, ya que la recolección es mucho mas sencilla y facilita el uso de maquinaria para la recogida de las cajas de uvas. 1.2b

Las empresa viticultora realiza campañas anuales para las vendimias. Estas campañas consisten en la ejecución de tareas específicas en todas las líneas de vides. En concreto, el grupo de tareas esenciales en cada linea para la realización de una vendimia son las siguientes:

Limpieza de las líneas: Consiste principalmente en la limpieza de los pasillos entre dos lineas de vides para facilitar las siguientes tareas. Normalmente, la tarea consiste en des-

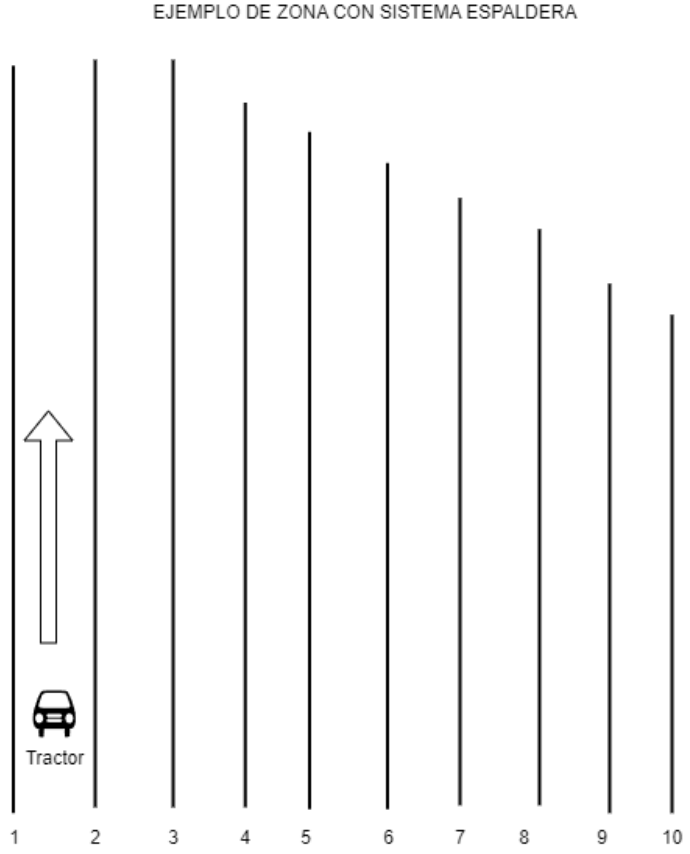
(a) Formación vid emparrado



(b) Maquinaria en emparrado

Figura 1.1: Sistema de formación Emparrado

(a) Formación vid emparrado



(b) Maquinaria en emparrado

Figura 1.2: Sistema de formación Espaldera

brozar, y suele ser personal propio de la empresa viticultora quien realiza esta tarea.

Poda de las líneas: Consiste en podar las hojas y ramas que estorban y que pueden entorpecer el trabajo de la recolección. En estas tareas trabajan conjuntamente capataces y personal de vendimia (personal externo) con experiencia previa.

Recolección de la uva: La tarea más significativa para la vendimia. Consiste en el corte de los racimos de uva y su almacenamiento en cajas. Estas cajas se llenan y se almacenan debajo de la estructura de la linea de la vid para facilitar la siguiente tarea de carga. En estas tareas, solo el personal de vendimia trabajara en las líneas, mientras que los empleados de la empresa supervisan el trabajo realizado (por lo que a partir de aquí se denominaran como capataces a los empleados de las vids). Estas son las más criticas y demanda muchos recursos.

Carga de las cajas de uva: Durante esta tarea, un capataz con permiso de conducción de tractor, junto con, normalmente, tres personas más de personal de vendimia, almacenarán las cajas de racimos de uva en el tractor, y cada vez que se llena el tractor o no queden cajas por recoger este será destinado a bodega donde se almacenaran los racimos de uva. Debe realizarse justo después de la recolección de la uva.

1.1 Motivación

Durante el inicio de la campaña anual de la vendimia, comenzando con las tareas de limpieza, la empresa no necesita muchos cambios en la organización, ya que el propio personal de la empresa realiza estas tareas. Sin embargo, a medida que avanza la campaña se incorpora mas gente, alcanzando el máximo en la tarea de recolección de la uva, por lo que la situación se vuelve caótica casi de un día para otro. Durante la etapa de recolección pueden ocurrir numerosos incidentes que provoquen la pausa en la recogida de la uva, como climatología adversa que afecte a todas las líneas. Sin embargo, incidentes específicos en líneas individuales, como lesiones de los operarios, picaduras de avispas, golpes de calor o fatiga que impida la finalización de los trabajos asignados, también son situaciones comunes. Evidentemente, los capataces atienden y socorren a los operarios de recolección en este tipo de situaciones. Sin embargo, después de atender a los operarios, a menudo pueden ocurrir desorganizaciones, como que el capataz no recuerde la linea donde se pauso la recolección, cambios incorrectos en los turnos de capataces que resultan en confusión sobre qué líneas supervisar, o que los propios capataces enfrenten urgencias personales que los obliguen a abandonar su puesto.

Normalmente, el administrador gestiona el avance de la campaña de vendimia comunicándose con los capataces mediante llamadas y aplicaciones de mensajería instantánea. Sin

embargo, pueden ocurrir incidencias no notificadas que pueden llevar a una mala organización de los recursos y a un seguimiento deficiente del progreso de los trabajos en las lineas.

Por tanto, la creación de una aplicación móvil para ayudar en la organización de las tareas y en el control del progreso de las lineas seria beneficiosa. Esto facilitaría la coordinación tanto para los capataces como para los administradores.

1.2 Objetivos

Los principales objetivos con la realización del proyecto son, poner orden al caos con:

1. Gestión de la información relativa a las líneas de vides, agrupadas por zonas. El administrador se encargará de decidir que líneas de vides son válidas para su posterior recolección.

2. Gestión de los empleados de la empresa viticultora(capataces). El administrador se encarga de añadir a los capataces en el sistema, los cuales podrán hacer uso de la aplicación.

3. Gestión de los operarios de vendimia. El administrador también es responsable de llevar un registro de la disponibilidad horaria de los operarios de vendimia, dado que puede variar considerablemente día a día. Además, la aplicación permitirá tomar asistencia del personal de vendimia para eliminar su presencia del calendario en caso de ausencia ese mismo día, o incluso dar de baja de manera permanente.

4. Gestión de las tareas de las líneas. Los capataces y los tractoristas son responsables de registrar el inicio de las tareas y de asignar los operarios de vendimia. Además, serán ellos quienes registren la finalización de las tareas, incluyendo el porcentaje de avance y cualquier incidente ocurrido, permitiendo así un seguimiento completo de la vendimia.

5. Idear funcionalidades que agilicen todo lo posible el proceso de seguimiento de las tareas.

Además de los puntos anteriores, mis objetivos también son mejorar mis conocimientos de Spring Framework, iniciarme en el desarrollo de aplicaciones móviles con Flutter y documentarme y experimentar con la generación automática de código y documentación mediante el uso de la especificación OpenAPI.

Capítulo 2
Tecnologías

En este capítulo expongo las tecnologías y herramientas utilizadas en la realización del proyecto.

2.1 Lenguajes de programación

2.1.1 JPQL

Jakarta Persistence Query Languaje (anteriormente Java Persistence Query Languaje) [3] es un lenguaje simple, basado en cadenas, similar a SQL, y que se utiliza para consultar entidades y sus relaciones.

2.1.2 Java

Java es un lenguaje de programación basado en clases y orientado a objetos. El motivo por el cual elegí este lenguaje es por la familiaridad con el mismo.

2.1.3 Yaml

YAML [4] es un lenguaje de serialización de datos diseñado para ser leído y escrito por humanos. Uso este lenguaje para crear archivos de configuración y especificación.

2.1.4 Dart

Dart [5] es un lenguaje de programación orientado a objetos y de código abierto desarrollado por Google. Hago uso de este lenguaje porque es el lenguaje de las aplicaciones de Flutter.

2.1.5 Moustache

Moustache [6] es un lenguaje para la creación de sistemas de plantillas. En concreto lo usaré para la modificación de configuraciones.

2.2 Librerías y/o Frameworks

2.2.1 Spring Framework

Spring Framework [7] es un marco de trabajo para el desarrollo de aplicaciones Java, que proporciona infraestructura para la gestión de dependencias, programación orientada a aspectos, seguridad, y soporte para aplicaciones web. Facilita la creación de aplicaciones robustas y escalables mediante su enfoque modular y su amplia gama de componentes integrados. Permite el uso de sus módulos, los cuales facilitan la construcción de la capa modelo y capas de controlador de la interfaz web. Las características mas importantes que ofrece Spring Framework en este proyecto son:

Inversión de Control(IoC) : Gestiona las dependencias de los objetos mediante la inyección de dependencias, facilitando la creación de aplicaciones desacopladas y más fáciles de probar.

Acceso a Datos Simplificado : Proporciona integración simplificada con diversas tecnologías de acceso a datos, como JDBC, JPA y Hibernate, a través de plantillas y abstracciones.

Desarrollo de Aplicaciones Web : Ofrece un robusto marco MVC (Model-View-Controller) para el desarrollo de aplicaciones web, incluyendo soporte para RESTful web services y WebSocket.

2.2.2 OpenAPI Specification/OAS

La especificación OpenAPI [8], anteriormente conocido como Swagger, que permite describir, producir, consumir y visualizar APIs HTTP. Esta especificación, que se describe en formato YAML o JSON, se usa para crear documentación automática de estas APIs y puede incluso mediante el uso la herramienta OpenAPI Generator, generar librerías de código con las que podremos dar uso tanto en lenguaje servidor como cliente.

2.2.3 Flutter

Flutter [9] es un framework de código abierto creado por Google para desarrollar aplicaciones nativas multiplataforma desde una única base de código. Utiliza el lenguaje de pro-

gramación Dart y permite la creación de aplicaciones móviles, web y de escritorio con alto rendimiento e interfaces de usuario personalizables.

2.3 Herramientas

2.3.1 Organización

Taiga

Taiga [10] es una herramienta gratuita y de código abierto orientada a la gestión de proyectos kaban y scrum.

2.3.2 Diseño

PlantUML

PlantUML [11] es un componente que permite crear diagramas UML a través de descripciones textuales simples. PlantUML provee una forma fácil de crear representaciones visuales de sistemas complejos.

Figma

Figma [12] es una herramienta de generación de prototipos. Se usará esta herramienta para la creación de prototipos de las pantallas del aplicativo móvil.

2.3.3 Bases de datos

H2

H2 [13] es una base de datos Java en memoria y muy liviana, con una interfaz fácil de utilizar en navegador. Se utilizara esta base de datos para el desarrollo de la aplicación.

MySQL

MySQL [14] es una base de datos relacional de código abierto, es una de las más populares gracias a su facilidad de uso, rendimiento y fiabilidad. Esta base de datos sería la utilizada en un entorno de producción.

2.3.4 Entornos de desarrollo(IDEs)

IntelliJ IDEA

IntelliJ IDEA [15] es un entorno de desarrollo creado por JetBrain para el desarrollo software, especialmente enfocado en el desarrollo de aplicaciones Java, pero también tiene soporte para otros lenguajes de programación.

Android Studio

Android Studio [16] es un entorno de desarrollo oficial de aplicaciones Android. Basado en IntelliJ IDEA, pero con muchas más funciones para la productividad del desarrollo móvil. También tiene soporte para el desarrollo de aplicaciones móviles con Flutter.

2.3.5 Control de versiones

Github

Github [17] plataforma de desarrollo colaborativo basada en Git [18]. Permite gestionar y compartir código, colaborar en proyectos, realizar seguimientos de cambios y revisar ramas mediante pull requests. Además, se hace uso de Github Actions para la realización de pruebas de integración e inspección continua.

2.3.6 Docker

Docker [19] es una plataforma de contenedorización que permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores portátiles y ligeros. Estos contenedores pueden ejecutarse de manera consistente en cualquier entorno, facilitando el despliegue y la escalabilidad de aplicaciones.

Capítulo 3

Metodologias

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos que se emplean para estructurar, planificar y controlar el proceso de desarrollo de software. En segundo lugar, filosofías o políticas que pueden analizar el modo en el que pretendemos adaptar tecnología con la definición de soluciones y que pueden dar lugar al desarrollo de las herramientas y técnicas que pueden elaborar una solución.

3.1 Metodologías ágiles

Las metodologías ágiles se fundamentan en el manifiesto ágil, creado en 2001 por un grupo de desarrolladores de software que buscaban una forma más efectiva de desarrollar el software, el cual tiene como principios:

1. **Individuos e interacciones sobre procesos y herramientas**: valorar más a los individuos y sus interacciones que a los procesos y las herramientas.
2. **Software funcionando sobre documentación extensiva**: el objetivo principal es proporcionar software funcional.
3. **Colaboración con el cliente sobre negociación contractual**: la comunicación constante con el cliente es esencial para asegurar el producto final.
4. **Respuesta ante el cambio sobre seguir un plan**: la capacidad de adaptarse y responder a los cambios es mejor que seguir un plan fijo.

Normalmente las metodologías ágiles se basan en un desarrollo incremental, lo cual se traduce en un software entregable que puede ser explotado. Este tipo de desarrollos nos permiten entregar pocas funcionalidades, pero evidentes y fácilmente adaptables.

Las metodologías ágiles permiten a los equipos de desarrollo adaptarse rápidamente a los cambios en los requisitos y prioridades del cliente, mejorar la colaboración y comunicación y entregar productos de calidad de manera continua.

Se explicará en detalle las metodologías más importantes, Scrum y Kanban.

3.1.1 Scrum

Framework de desarrollo ágil con el fin de administrar proyectos complejos. Fomenta la colaboración, flexibilidad y la entrega incremental de productos y centrado en la mejora continua.

Scrum está concebido con el propósito de asistir a los equipos a adaptarse de manera natural a las circunstancias y exigencias de los usuarios, mediante el cambio de prioridades integrado en el proceso y los ciclos cortos para que el equipo pueda adquirir y mejorar constantemente.

Detallaremos los roles de un equipo de Scrum y sus responsabilidades, los artefactos que definen el producto y el trabajo que hay que hacer para crear el producto.

Roles:

1. **Product Owner:** es el representante del cliente, y se centra en entender los requisitos empresariales de los clientes y mercado.
2. **Scrum Master:** facilita el proceso de Scrum, ayuda a eliminar obstáculos y asegura que el equipo siga las prácticas de Scrum.
3. **Equipo de desarollo:** grupo de desarrolladores que trabajan en los incrementos del producto.

Eventos:

1. **Sprint:** un ciclo de trabajo corto y repetitivo, típicamente de 2 a 4 semanas en el que se crea un incremento del producto utilizable.
2. **Planificación de Sprints:** reunión dirigida por el Scrum Master donde el equipo decide el objetivo y alcance del Sprint actual. Además, se añaden al Sprint historias de usuario especificas a partir del backlog del producto.
3. **Scrum diario:** reunión diaria, en la que el equipo lleva a cabo una planificación temporal de las actividades correspondientes para las siguientes 24 horas.
4. **Revisión del Sprint:** reunión final del Sprint en el que el equipo presenta el resultado al Product Owner.

Artefactos:

1. **Product backlog:** se trata de la lista de tareas principales que se deben llevar a cabo. debe mantener el Product Owner a medida que crece y evoluciona durante el desarrollo del producto. A estos trabajos se les denomina historias de usuario.
2. **Sprint Backlog:** se trata de una lista de elementos e historias de usuario seleccionadas por el equipo de desarrollo con el fin de implementarlos en el Sprint actual.
3. **Incremento:** la suma de todos los elementos del producto backlog completados durante el sprint actual.
4. **Épicas:** es un conjunto de trabajo grande que puede dividirse en tareas específicas.

3.1.2 Kanban

Esta metodología ágil es utilizada para gestionar y mejorar el trabajo. Requiere una comunicación en tiempo real sobre la capacidad y una total transparencia del trabajo.

Es una forma de visualizar el trabajo y limitar la cantidad de trabajo en curso para que el equipo pueda lograr un flujo de trabajo eficiente.

En esta metodología se hace uso de un tablero de Kanban, que se utiliza para visualizar el flujo de trabajo entre los equipos y garantiza que se visualice el trabajo realizado, estandarice su flujo de trabajo, y se identifiquen y resuelvan todos los impedimentos. Estos tableros tienen un flujo básico de tres pasos:

1. **Por hacer (To do):** tareas pendientes.
2. **En progreso (In progress):** tareas que se están realizando en un momento determinado.
3. **Hecho (Done):** tareas finalizadas.

3.1.3 Scrumban

La metodología Scrumban [20] combina las mejores características de Scrum y Kanban en un framework de gestión de proyectos híbrido. Utiliza el flujo de trabajo visual del Kanban y la estructura estable de Scrum.

Scrum contribuye a Scrumban en los elementos principales de Sprints y reuniones rápidas diarias.

Kanban contribuye a Scrumban con el tablero de Kanban y las tarjetas representantes de las tareas del proyecto y su estado.

3.1.4 Metodología escogida

En el proyecto se utiliza la metodología ágil Scrumban mediante el uso de las herramientas que nos proporcionan las metodologías anteriores, Kanban y Scrum.

Utilizaremos una herramienta del tablero de Kanban digital, para la cual hay una gran variedad de herramientas disponibles. Se optó por Taiga ya que es una herramienta Open-Source enfocada en el uso de elementos Kanban y Scrum.

Además del tablero Kanban digital, también se hace uso de las historias de usuario derivadas de las funcionalidades a implementar especificadas durante el análisis, así como las épicas.

El proyecto se estructura en distintos Sprints, en los que se realizarán entre 1 o 2 épicas en función de la cantidad de historias de usuario. Se realizará una evaluación del diseño al comienzo de cada Sprint, aunque el primer Sprint consistirá en el diseño de una primera versión de la capa modelo y servicios, y también se elaborarán prototipos de las pantallas del aplicativo móvil.

Dado que se trata de un proyecto llevado a cabo por una persona no se podrán aplicar ciertas prácticas de colaboración entre miembros de un equipo de desarrollo. Sin embargo, se empleará un sistema de puntuación que proporciona la herramienta para asignar "puntos historia" a cada una de las historias de usuario, lo que nos permitirá visualizar el nivel de esfuerzo requerido para la ejecución de las tareas.

Capítulo 4
Análisis

En esta sección se especificarán los actores que intervienen en el funcionamiento de la aplicación, las historias de usuario y los Sprints que se realizarán.

4.1 Actores

Se distinguen tres actores principales que utilizan de la aplicación:

4.1.1 Administrador

Se encargará de administrar la información de la empresa viticultora, comenzando por el registro de los empleados del viñedo, quienes, tal como se ha explicado previamente, mantienen y conservan las vides durante el transcurso del año. No obstante, durante la vendimia desempeñan los roles de capataces y tractoristas.

Además del registro de los empleados del viñedo, también se ocupará de la contratación y registro de los operarios de la vendimia y sus horarios, ya que se contratan por hora trabajada, por lo que es imprescindible registrar la asistencia de cada uno de los operarios.

El administrador también se encarga del registro y actualización de las líneas de vides, es decir, registrará toda la información relativa a la vid, tipo de vid, longitud de la línea, año de planta, si es apta para la recolección, etc.

También tiene la responsabilidad de supervisar las fases de la campaña de vendimia, es decir, iniciar las fases de limpieza, poda y recolección.

4.1.2 Capataz

Durante el transcurso de la campaña, el capataz se encargará de iniciar las tareas pendientes en cada una de las fases de la campaña. Para iniciar una tarea, el capataz asigna operarios a las líneas en las que hay una tarea pendiente, comunicándolo al sistema. Si la tarea finaliza o se pausa, el capataz se encarga de notificarlo al sistema, comentando el trabajo realizado en

esa línea y anotando el porcentaje de progreso del sistema. Además, al completar o pausar estos trabajos realizados, el capataz decidirá si se debe notificar al tractorista para que recoja las cajas con racimos de uvas.

4.1.3 Tractorista

Existen algunos capataces que se especializan en el rol de tractorista. Durante la fase de recolección, los tractoristas se encargan de recoger con el tractor las cajas de racimos de uva recolectadas por los operarios. Debido a que la uva debe ser enviada a bodega con la mayor brevedad posible, los tractoristas recibirán notificaciones cuando se generen tareas pendientes para la recogida. El tractorista también se ocupará de asignar operarios para la recogida de estas cajas.

4.2 Historias de Usuario

A continuación, se detallan todas las historias de usuario, agrupadas en épicas, tanto las historias de usuario para el desarrollador como para los roles específicos de nuestra aplicación. Las historias tienen asignado un identificador y una estimación en puntos historia. Esta lista de historias de usuario fue modificada a medida que evolucionaba el proyecto, principalmente en las revisiones de cada inicio de los Sprints.

En la primera épica "Iniciación de proyecto y Autenticación de Usuario", agrupa las historias de usuario relacionadas con la autenticación, registro de nuevos usuarios y modificación de datos personales y contraseña.

ID	Historia de Usuario	Puntos Historia
1	*Como usuario, quiero poder iniciar sesión en la aplicación*	48
2	*Como usuario, quiero poder cerrar sesión en la aplicación*	8
3	*Como usuario, quiero cambiar la contraseña de acceso a la aplicación*	20
4	*Como administrador quiero registrar nuevos usuarios y asignarles roles de capataz y tractorista*	18
5	*Como usuario quiero modificar mis datos de usuario*	16

Tabla 4.1: Historias de usuario de la épica "Iniciación de proyecto y Autenticación de Usuario"

En esta épica "Trabajadores y Horarios" se agrupan las historias de usuario encargadas del registro de operarios de vendimia y sus respectivos calendarios por parte de los administradores.

ID	Historia de Usuario	Puntos Historia
6	*Como administrador, quiero poder registrar a operarios de vendimia*	10
7	*Como administrador, quiero obtener información de los operarios, resumida e individual*	10
8	*Como administrador, quiero poder actualizar la información de un operario de vendimia*	10
9	*Como administrador quiero poder dar de baja a un operario de vendimia*	10
10	*Como administrador, quiero visualizar el calendario de disponibilidad de los operarios*	10
11	*Como administrador, quiero actualizar las horas de una fecha del calendario de un operario*	10
12	*Como administrador, quiero visualizar un listado de operarios y sus horarios actuales, para poder registrar su presencialidad*	16

Tabla 4.2: Historias de usuario de la épica "Trabajadores y Horarios"

En la épica "Gestión de Zonas y Líneas", tenemos el conjunto de historias de usuario encargadas de ofrecer las funcionalidades de gestión de la información relativa a líneas y sus respectivas zonas.

ID	Historia de Usuario	Puntos Historia
13	*Como administrador, quiero añadir zonas de líneas de vides*	8
14	*Como administrador, quiero visualizar la lista de zonas de vides*	8
15	*Como administrador, quiero ver la información de una zona específica*	8
16	*Como administrador, quiero modificar la información de una zona específica*	6
17	*Como administrador, quiero añadir una línea a una zona*	6
18	*Como administrador quiero ver la lista de líneas pertenecientes a una zona específica*	10
19	*Como administrador, quiero obtener la información de una línea*	6
20	*Como administrador, quiero modificar la información de una línea*	6
21	*Como administrador, quiero poder deshabilitar/habilitar una línea para su futura recolección*	4
22	*Como administrador, quiero eliminar una línea de una zona específica*	6

Tabla 4.3: Historias de usuario de la épica "Gestión de Zonas y Líneas"

Esta pequeña épica "Campaña" se agrupan las historias de usuario relacionadas con la gestión de la campaña anual por parte del administrador, y la generación de tareas iniciales en cada fase de acuerdo con la información de las líneas.

ID	Historia de Usuario	Puntos Historia
23	*Como administrador, quiero iniciar la campaña, para que se generen las tareas pendientes de limpieza*	8
24	*Como administrador, quiero iniciar la fase de poda, para que se generen las tareas pendientes de poda*	6
25	*Como administrador, quiero iniciar la fase de recolección, para que se generen las tareas pendientes de recolección*	5
26	*Como administrador, quiero poder finalizar la campaña anual*	2

Tabla 4.4: Historias de usuario de la épica "Campaña"

La épica ”Tareas de Capataz” se compone de las historias de usuario que ayudan al capataz al progreso adecuado de la fase de campaña actual, pudiendo gestionar y hacer seguimiento de las tareas y los operarios.

ID	Historia de Usuario	Puntos Historia
27	*Como capataz, quiero poder visualizar las tareas pendientes, con la información de zona y línea*	7
28	*Como capataz, quiero ver los detalles de una tarea pendiente*	5
29	*Como capataz, quiero poder iniciar tareas, con o sin operarios*	7
30	*Como capataz, quiero visualizar la lista de las tareas en progreso*	6
31	*Como capataz, quiero poder ver en detalle una tarea en progreso*	4
32	*Como capataz, quiero poder finalizar una tarea en progreso*	5
33	*Como capataz, quiero visualizar la lista de las tareas finalizadas*	5
34	*Como capataz, quiero ver en detalle una tarea finalizada*	4
35	*Como capataz, quiero filtrar las tareas pendientes por zona y línea*	5
36	*Como capataz, quiero filtrar por zona y línea mediante un lector de códigos QR para identificar una línea específica*	5

Tabla 4.5: Historias de usuario de la épica “Tareas de Capataz”

Esta épica ”Tareas de Tractorista” es muy similar a la épica ”Tareas de capataz”, pero esta épica agrupa historias de usuario que ofrecen funcionalidades específicas al tractorista para una mejor gestión de tareas, permitiendo una mayor agilidad en la iniciación y finalización de tareas.

ID	Historia de Usuario	Puntos Historia
37	*Como capataz, quiero, al finalizar una tarea, indicar si hace falta notificar al tractorista, para generar una nueva tarea de tractorista*	6
38	*Como tractorista, quiero visualizar el listado de tareas pendientes*	5
39	*Como tractorista, quiero poder seleccionar múltiples tareas específicas de tractorista, para iniciarlas con también múltiples operarios*	7
40	*Como tractorista, quiero visualizar todas las tareas en progreso específicas de tractorista*	4
41	*Como tractorista, quiero ser notificado cuando se generen nuevas tareas específicas de tractorista*	4

Tabla 4.6: Historias de usuario de la épica “Tareas de Tractorista”

4.3 Sprints

El proyecto se ha estructurado en 5 Sprints. A pesar de que la duración habitual de los Sprints oscila entre dos y cuatro semanas en función de la cantidad de historias de usuario a realizar, los dos primeros Sprints se llevaron a cabo en dos meses cada uno. En estos Sprints adquirimos conocimiento acerca de cómo usar OpenAPI, personalizar esta generación de clases y adaptarnos al desarrollo de aplicaciones móviles mediante Flutter.

4.3.1 Sprint 1

En este Sprint inicial se realizan la inicialización del proyecto con una estructura básica para el registro y autenticación de usuarios mediante JWT, y el uso de OpenAPI codegen y Swagger. Esto equivale a la implementación de las historias de usuario de dos épicas:

ID	Historia de Usuario	Puntos Historia
1	*Como usuario, quiero poder iniciar sesión en la aplicación*	48
2	*Como usuario, quiero poder cerrar sesión en la aplicación*	8
3	*Como usuario, quiero cambiar la contraseña de acceso a la aplicación*	20
4	*Como administrador quiero registrar nuevos usuarios y asignarles roles de capataz y tractorista*	18
5	*Como usuario quiero modificar mis datos de usuario*	16

Tabla 4.7: Historias de usuario de la épica “Iniciación de proyecto y Autenticación de Usuario”

Este Sprint es el más denso en puntos historia, ya que no se conocen en detalle las herramientas OpenAPI y Swagger y cómo usarlas de forma eficaz, por lo que requiere de más esfuerzo.

4.3.2 Sprint 2

En el Sprint 2 se implementa el registro de operarios de vendimia y sus calendarios. El objetivo de este Sprint es implementar las historias de usuario de la épica de trabajadores y horarios.

ID	Historia de Usuario	Puntos Historia
6	*Como administrador, quiero poder registrar a operarios de vendimia*	10
7	*Como administrador, quiero obtener información de los operarios, resumida e individual*	10
8	*Como administrador, quiero poder actualizar la información de un operario de vendimia*	10
9	*Como administrador quiero poder dar de baja a un operario de vendimia*	10
10	*Como administrador, quiero visualizar el calendario de disponibilidad de los operarios*	10
11	*Como administrador, quiero actualizar las horas de una fecha del calendario de un operario*	10
12	*Como administrador, quiero visualizar un listado de operarios y sus horarios actuales, para poder registrar su presencialidad*	16

Tabla 4.8: Historias de usuario de la épica "Trabajadores y Horarios"

Este Sprint sigue siendo denso en puntos historia, dado que no se conoce bien cómo trabajar con el framework de Flutter, y la integración de generación de código con la herramienta OpenAPI en la capa cliente. Este Sprint es sumamente importante para el proyecto, ya que se encarga de implementar las funcionalidades más relevantes para el administrador y el progreso adecuado de las campañas, tales como la gestión de información de los operarios, sus calendarios, y la funcionalidad de "pasar lista" a los operarios.

4.3.3 Sprint 3

Este Sprint se centra en la implementación las funcionalidades de administrador de gestión de la información de líneas de vides agrupadas por zonas. Estas funcionalidades son sumamente relevantes para la empresa, ya que gestiona la información de uno de los activos más relevantes, las vides.

Para este Sprint se asigna todas las historias de usuario de la épica "Gestión de líneas"

ID	Historia de Usuario	Puntos Historia
13	*Como administrador, quiero añadir zonas de líneas de vides*	8
14	*Como administrador, quiero visualizar la lista de zonas de vides*	8
15	*Como administrador, quiero ver la información de una zona específica*	8
16	*Como administrador, quiero modificar la información de una zona específica*	6
17	*Como administrador, quiero añadir una línea a una zona*	6
18	*Como administrador quiero ver la lista de líneas pertenecientes a una zona específica*	10
19	*Como administrador, quiero obtener la información de una línea*	6
20	*Como administrador, quiero modificar la información de una línea*	6
21	*Como administrador, quiero poder deshabilitar/habilitar una línea para su futura recolección*	4
22	*Como administrador, quiero eliminar una línea de una zona específica*	6

Tabla 4.9: Historias de usuario de la épica "Gestión de Zonas y Líneas"

A partir de este Sprint, se reduce la cantidad de puntos historia, lo que a su vez reduce la duración de los Sprints.

4.3.4 Sprint 4

En este Sprint se alcanzaron las funcionalidades que por las que nace el proyecto, que son las derivadas de la necesidad de gestión de las tareas durante la campaña. En este Sprint se realizan dos épicas. En primer lugar, la épica relacionada con la gestión de las fases de la campaña anual y la generación de tareas pendientes asociadas a cada fase y línea. La segunda épica "Tareas de capataz", relacionada con la gestión de las tareas por parte de los capataces, es decir, iniciarlas y finalizar las tareas, asignándoles operarios, y registrar información sobre el progreso de las líneas. Además, para simplificar el uso de la aplicación por parte de los capataces, se implementa la funcionalidad de leer los códigos QR para identificar las líneas, con el fin de filtrar las tareas pendientes de esa línea y comenzarlas mucho más rápidamente.

ID	Historia de Usuario	Puntos Historia
23	*Como administrador, quiero iniciar la campaña, para que se generen las tareas pendientes de limpieza*	8
24	*Como administrador, quiero iniciar la fase de poda, para que se generen las tareas pendientes de poda*	6
25	*Como administrador, quiero iniciar la fase de recolección, para que se generen las tareas pendientes de recolección*	5
26	*Como administrador, quiero poder finalizar la campaña anual*	2

Tabla 4.10: Historias de usuario de la épica "Campaña"

ID	Historia de Usuario	Puntos Historia
27	*Como capataz, quiero poder visualizar las tareas pendientes, con la información de zona y línea*	7
28	*Como capataz, quiero ver los detalles de una tarea pendiente*	5
29	*Como capataz, quiero poder iniciar tareas, con o sin operarios*	7
30	*Como capataz, quiero visualizar la lista de las tareas en progreso*	6
31	*Como capataz, quiero poder ver en detalle una tarea en progreso*	4
32	*Como capataz, quiero poder finalizar una tarea en progreso*	5
33	*Como capataz, quiero visualizar la lista de las tareas finalizadas*	5
34	*Como capataz, quiero ver en detalle una tarea finalizada*	4
35	*Como capataz, quiero filtrar las tareas pendientes por zona y línea*	5
36	*Como capataz, quiero filtrar por zona y línea mediante un lector de códigos QR para identificar una línea específica*	5

Tabla 4.11: Historias de usuario de la épica “Tareas de Capataz”

4.3.5 Sprint 5

En este último Sprint se centra en implementar las tareas específicas para el tractorista, las cuales ocurren en la fase de recolección de la uva y deben realizarse inmediatamente después de las tareas de recolección, ya sea porque se recolecta la línea en su totalidad o porque se pausó la recolección. El capataz decide notificar al tractorista para que proceda a la recogida de las cajas de la uva de una línea. El tractorista podrá seleccionar múltiples tareas simultáneamente, ya que son tareas mucho más rápidas que las de recolección, lo que facilitaría el uso de la aplicación al registrar los trabajos realizados. Para este Sprint se realizarán todas las tareas pendientes del backlog, es decir, todas las tareas de la épica "Tareas de Tractorista".

ID	Historia de Usuario	Puntos Historia
37	*Como capataz, quiero, al finalizar una tarea, indicar si hace falta notificar al tractorista, para generar una nueva tarea de tractorista*	6
38	*Como tractorista, quiero visualizar el listado de tareas pendientes*	5
39	*Como tractorista, quiero poder seleccionar múltiples tareas específicas de tractorista, para iniciarlas con también múltiples operarios*	7
40	*Como tractorista, quiero visualizar todas las tareas en progreso específicas de tractorista*	4
41	*Como tractorista, quiero ser notificado cuando se generen nuevas tareas específicas de tractorista*	4

Tabla 4.12: Historias de usuario de la épica "Tareas de Tractorista"

Capítulo 5

Diseño

En esta sección se explicará la arquitectura elegida para el desarrollo de la aplicación.

5.1 Arquitectura

Para el diseño de la aplicación, denominada *Harvest*, nos basaremos en una arquitectura cliente-servidor, en la cual la comunicación se llevara a cabo mediante el protocolo HTTP. El cliente será un aplicativo móvil y el servicio un aplicativo web REST.

La esta arquitectura cliente-servidor, consistirá en un aplicativo web REST que da servicio a un frontend, en nuestro caso un aplicativo móvil. Este modelo de diseño es ampliamente empleado para la creación de aplicaciones distribuidas.

5.2 Capa OpenAPI

Además de las capas frontend y backend quiero introducir la capa OpenAPI en esta sección, la cual nos permitirá aplicar un desarrollo *API-First* [21]. API-First es un enfoque de desarrollo de software en el cual la API es diseñada y definida antes de comenzar a desarrollar cualquier parte del sistema, convirtiéndose en el contrato que guía el desarrollo de los componentes backend y frontend. La aproximación de API-First nos aporta los siguientes beneficios:

1. Los equipos de desarrollo pueden trabajar en paralelo al establecer un ”contrato” del API. Será especialmente útil para definir las interfaces de controlador del API y seguir la estructura establecida por el desarrollo backend. Para el desarrollo frontend, también podemos probarlo invocando una API de prueba, sin tener que esperar a una implementación completa de la API.

2. Reduce el costo de desarrollo de apps. El diseño API-First posibilita la reutilización de APIs en múltiples proyectos.

3. Aumenta el ”speed to market”. Automatizando muchos procesos de desarrollo de APIs, pudiendo generar documentación de la API de manera automática con Swagger [22], generar y desplegar APIs de prueba[23].

4. Asegura una buena experiencia de desarrollador, garantizando que los desarrolladores tengan siempre una API bien diseñada y documentada, lo cual reduce la curva de aprendizaje de los desarrolladores para usar una API.

5. Reduce el riesgo de fallo. Asegurando que las APIs son fiables , consistentes y fáciles de usar.

Para la elaboración del proyecto se utilizarán dos herramientas específicas para el desarrollo:

1. OpenAPI codegen [24]: herramienta para la generación automática de código fuente a partir de la especificación OpenAPI, en este caso OAS 3 [25], define cómo debe estructurarse una API REST, describiendo endpoints, métodos HTTP, parámetros, respuestas, esquemas de datos, autenticación, etc. Esta especificación se presenta en un formato estandarizado en JSON o Yaml. En nuestro caso, usaré la herramienta para generar código fuente en dos tipos de lenguaje diferentes, Java-Spring y Dart, como servidor y cliente, respectivamente.

2. SpringDoc [26]: herramienta para la generación automática de documentación que usa proyectos spring boot. Funciona examinando la aplicación en tiempo de ejecución para inferir semánticas API basadas en la configuración de Spring, estructura de clase y varias anotaciones.

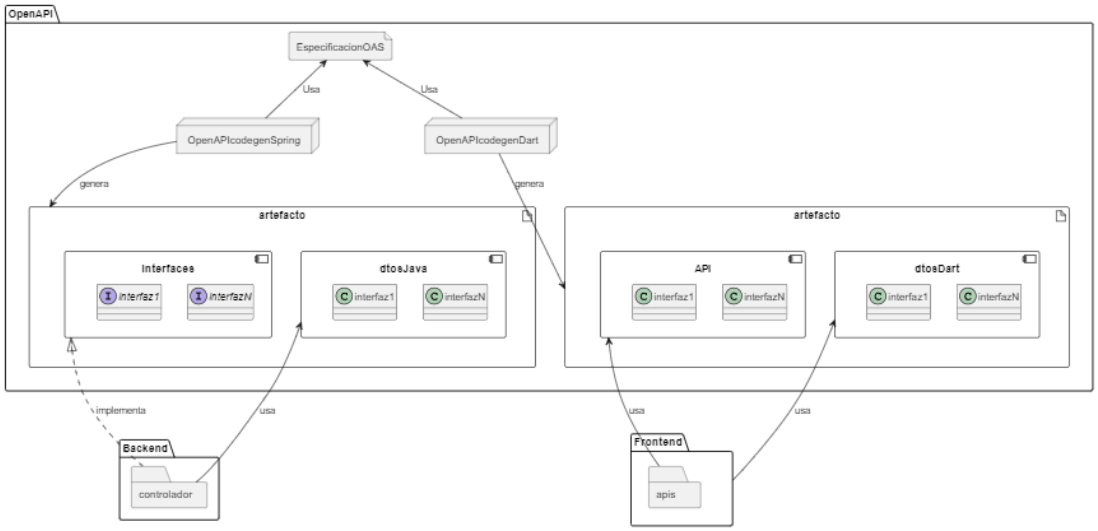


Figura 5.1: Esquema OpenAPI

Este diagrama 5.1 ilustra cómo, mediante el uso de la especificación, se emplea la herramienta de OpenAPI Codegen para la creación de DTOs e interfaces del controlador, que posteriormente se implementarán en la capa REST del backend. También se vuelve a usar la herramienta para implementar la capa de acceso a datos y DTOs del frontend, simplificando enormemente el desarrollo.

5.3 Backend

En esta sección se presentará de manera detallada el diseño de las capas del backend 5.2 siguiendo un desarrollo APIFirst. Cabe destacar que, antes de iniciar este tipo de desarrollo, durante la primera iteración, se elaboró previamente el modelo para estructurar todas las clases y las relaciones entre ellas.

5.3.1 Capa Rest

La capa de presentación o controladores de una API REST es responsable de manejar las solicitudes HTTP que provienen del frontend u otros clientes. Aquí, se implementan las interfaces generadas por OpenAPI, asegurando que cada endpoint definido en la especificación tenga su correspondiente manejo en la aplicación. Los controladores validan y procesan las solicitudes, luego redirigen la información a los servicios correspondientes para que se ejecute la lógica de negocio.

Para gestionar mejor la gran cantidad de operaciones, estas se agrupan en diferentes controladores según su funcionalidad. Por ejemplo, un controlador puede encargarse de todas las operaciones relacionadas con usuarios y autenticación, mientras que otro maneja las relacionadas con las líneas. Esta división facilita el mantenimiento, ya que cada controlador maneja un conjunto específico de operaciones, lo que mejora la organización y escalabilidad de la aplicación.

A continuación, se enumeran los controladores que se han implementado:

1. AuthenticationController: Se encarga de las operaciones de registro, actualización e ingreso de usuarios.

2. TrabajadorController: Encargado del registro, actualización y bajas de los operarios y sus respectivos calendarios.

3. LineasController: Este controlador expone las operaciones de registro, actualización, y eliminación de líneas y zonas.

4. CampanhaController: Encargado de las operaciones de control de campaña y sus cambios de fase.

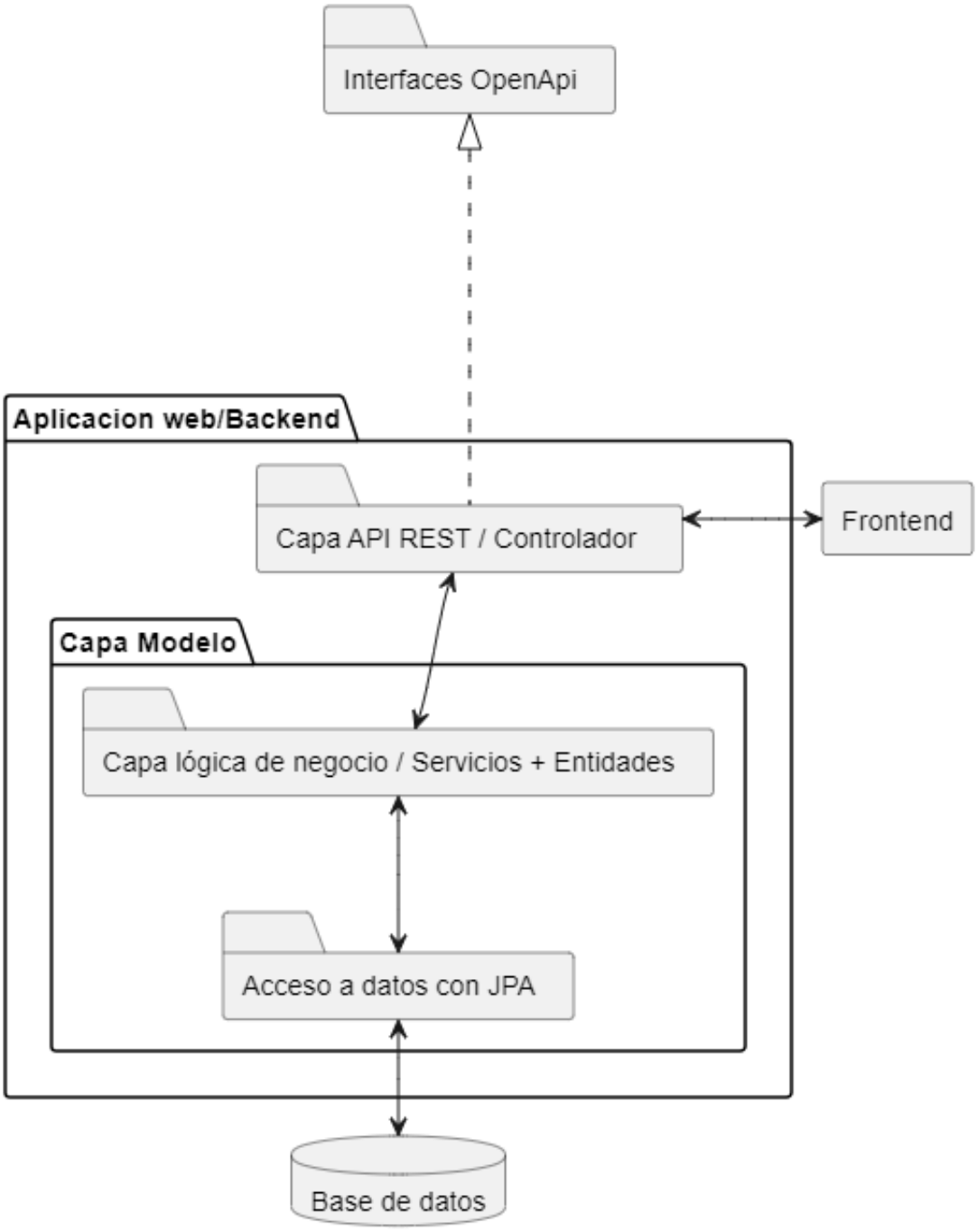


Figura 5.2: diagrama backend

5. CapatazController: Se encarga de las operaciones de obtención de tareas, tanto pendientes como las que están en progreso y finalizadas. Además, también cuenta con las operaciones de obtención de operarios disponibles para su asignación en la tarea.

6. TractoristaController: Las mismas operaciones de CapatazController, pero específicas para el rol de tractorista.

5.3.2 Capa Modelo

Generalmente, se considera que la capa de modelo está orientada a representar los datos y la lógica de negocio, la cual es contenida dentro de los servicios.

En esta capa se representan las entidades, que son objetos de dominio de la aplicación y que se mapean a las tablas en la base de datos. Esta capa es fundamental para determinar la estructura de los datos. A continuación, se representan estas entidades y sus relaciones 5.3 5.4.

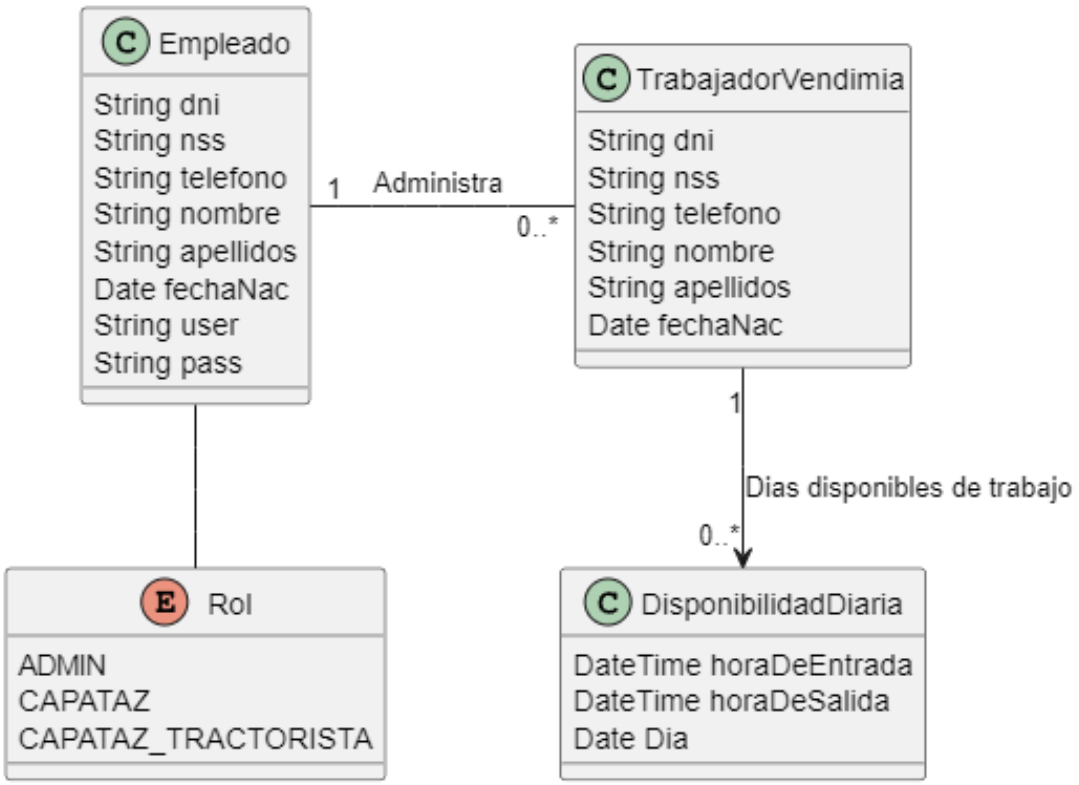


Figura 5.3: Diagrama de empleados y operarios

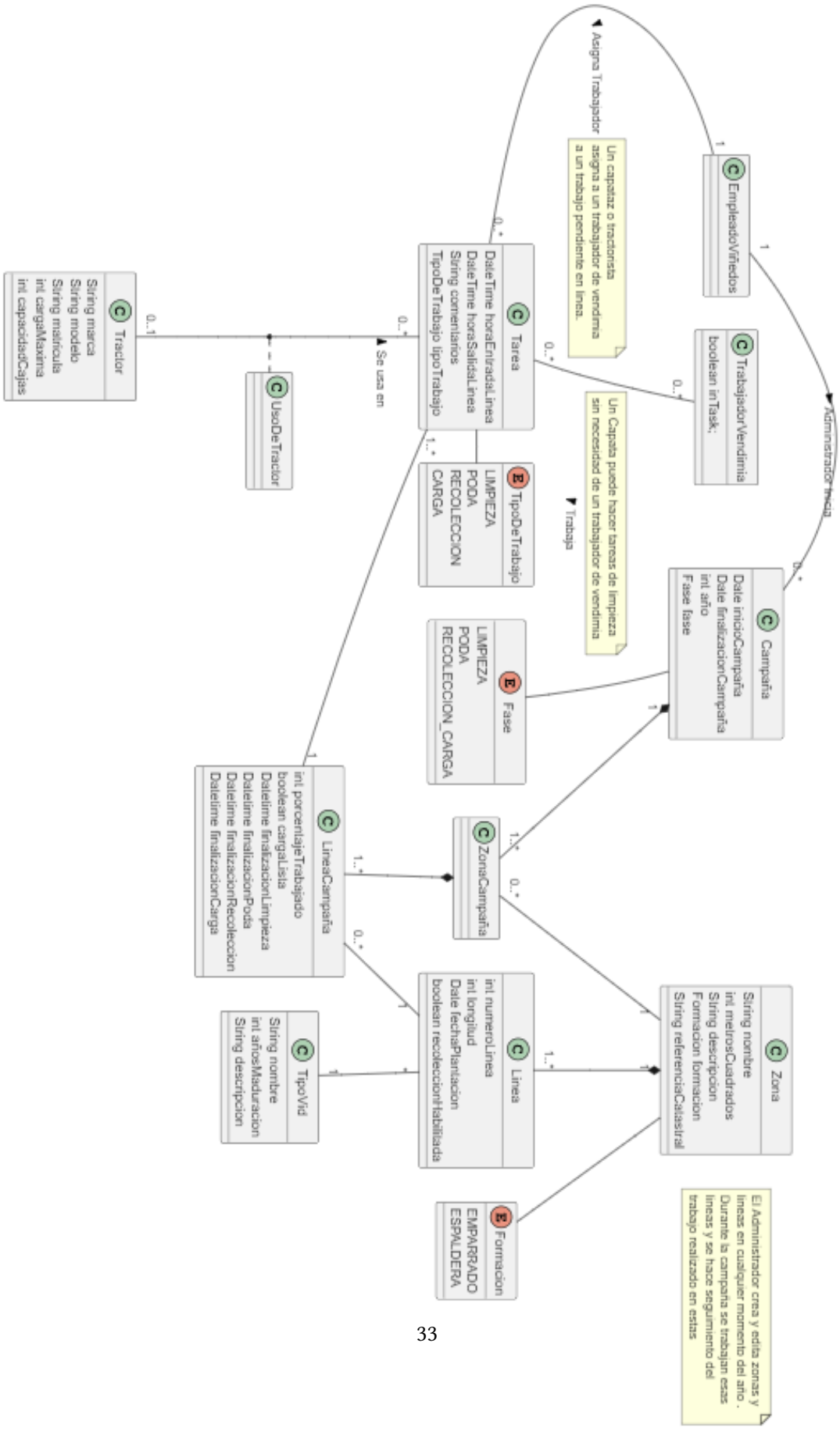


Figura 5.4: Diagrama de el resto del modelo

Servicios

La capa de servicios en una aplicación API REST encapsula la lógica de negocio, actuando como intermediaria entre la capa de controladores y la capa de modelo. Aquí, se procesan y coordinan las operaciones necesarias para cumplir con las solicitudes del usuario, como realizar cálculos, aplicar reglas de negocio, y orquestar el acceso a los datos a través de los repositorios. Esta capa permite mantener la lógica de negocio centralizada, organizada y desacoplada de los detalles de la interacción con la base de datos o de la gestión de las solicitudes HTTP.

Capa acceso a datos

5.4 Frontend

5.4.1 Capa acceso a servicios OpenAPI

5.4.2 Interfaz de usuario

Capítulo 6

Implementación

6.1 Implementación del Backend

6.2 Implementación del Frontend

Capítulo 7

Pruebas y herramientas de análisis de código

7.1 Herramientas de análisis

7.1.1 Sonar

7.1.2 Spotbugs

7.1.3 Jacoco

7.2 Pruebas de Unidad

7.2.1 Pruebas Mock

7.3 Pruebas de Integración

7.4 Pruebas de Aceptación

Capítulo 8

Planificación y evaluación de costes

Capítulo 9

Conclusiones

9.1 Puntos a mejorar

Mejor estructura del proyecto, un orden mejor para la agrupacion de la rest API, con versionado, paginacion,

9.1.1 Pruebas

Contar un poco de como podría haber hecho mas cobertura etc

9.1.2 Funcionalidades

Contar un poco de funcionalidades de bodega que no cubri en el TFG

Capítulo 10

Contido demostrativo

ENTRE a introdución e as conclusións, o documento conterá tantos capítulos como sexa preciso, sempre con coidado de non rebasar o límite de 80 páxinas fixado polo regulamento de TFGs.

Empregaremos éste de xeito demostrativo, para ilustrar o uso de elementos habituais que poidan ser de utilidade[1].

10.1 Inclusión de imaxes

Se precisamos imaxes no noso documento, incluirémolas do xeito que se indica na figura 10.1 (páxina 39). Se o facemos así, LaTeXubicará cada imaxe no mellor lugar posible, lugar que pode variar a medida que o documento vaia crecendo coa inclusión de máis texto e outros elementos (máis imaxes, táboas, etc.).

Figura 10.1: Pé de imaxe descritivo

Recoméndase almacenar os ficheiros gráficos no directorio `imaxes`.

10.1.1 Inclusión de varias sub-imaxes

Se precisamos inserir imaxes relacionadas, pode ser apropiado incluílas como sub-figuras, do xeito que se pode apreciar na figura 10.2 coas imaxes 10.2a e 10.2b. Como se pode ver nos exemplos desta sección, sempre é recomendable referirse ás imaxes pola súa referencia, xa que dese xeito non dependemos de onde queden ubicados os elementos en cuestión.

[1] Por exemplo, isto é unha nota a pé de páxina.

(a) Pé de subimaxe rotada

(b) Pé de subimaxe deformada

Figura 10.2: Pé de imaxe xeral

10.2 Inclusión de táboas

Se precisamos táboas no noso documento, incluirémolas do xeito que se indica na táboa 10.1 (páxina 41). Se o facemos así, LaTeXubicará cada táboa no mellor lugar posible, lugar que pode variar a medida que o documento vaia crecendo coa inclusión de máis texto e outros elementos (máis imaxes, táboas, etc.).

Título de columna	**Outro título de columna**
Título de fila	Contido de celda
Título de fila	Contido de celda
Título de fila	Contido de celda
Título de fila	Contido de celda
Título de fila	Contido de celda
Título de fila	Contido de celda

Tabla 10.1: Pé de táboa descritivo

Para táboas longas que ocupan varias páxinas, como é o caso da 10.2, recoméndase o uso do paquete `lontable`, descomentando a liña correspondente no ficheiro raíz do proxecto (`memoria_tfg.tex`).

Tabla 10.2: Pé descritivo dunha táboa longa

Primeira columna	**Segunda columna**	**Terceira columna**
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778

. . . . . *(continúa na páxina seguinte)* . . . . . .

Tabla 10.2 – *(vén da páxina anterior)*

Primeira columna	**Segunda columna**	**Terceira columna**
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778

. . . . *(continúa na páxina seguinte)* . . . .

Tabla 10.2 – *(vén da páxina anterior)*

Primeira columna	Segunda columna	Terceira columna
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778
Texto de exemplo	abcdef ghjijklmn	123.456778

10.3 Inclusión de código fonte

Se precisamos incluír fragmentos de código fonte, podemos facelo, por exemplo, da seguinte maneira:

```
#include <stdio.h>
#define N 10

int main()
{
  int i;

  // Isto é un comentario
  puts("Ola, mundo!");

  for (i = 0; i < N; i++)
  {
    puts("LaTeX é a ferramenta de edición ideal para profesionais
    da informática!");
  }

  return 0;
}
```

10.4 Uso da relación de acrónimos e do glosario

Os acrónimos edítanse no ficheiro `bibliografia/acronimos.tex` e úsanse empregando a orde `acrlong` para obter o termo completo (deste xeito: Erlang Open Telecom Platform), a orde `acrshort` para obter o acrónimo (deste xeito: ERLANG/OTP). A primeira vez que usamos un termo con acrónimo no documento é recomendable usar orde `acrfull` (que produce ambas versións á vez: Erlang Open Telecom Platform (ERLANG/OTP)). Os acrónimos que non se usan no documento, non aparecen na relación que se xerar na versión PDF.

Pola súa banda, os termos do glosario edítanse no ficheiro `bibliografia/glosario.tex` e úsanse empregando a orde `gls` (deste xeito, bytecode) ou `Gls` (deste xeito, Bytecode). Ao igual que os acrónimos, os termos que non se usan no documento, non aparecen na relación que se xera na versión PDF.

Apéndices

Apéndice A

Material adicional

Exemplo de capítulo con formato de apéndice, onde se pode incluír material adicional que non teña cabida no corpo principal do documento, suxeito á limitación de 80 páxinas establecida no regulamento de TFGs.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque.

Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lista de acrónimos

ERLANG/OTP Erlang Open Telecom Platform. 44

Glosario

bytecode Código independente da máquina que xeran compiladores de determinadas linguaxes (Java, Erlang,...) e que é executado polo correspondente intérprete.. 44

Bibliografía

[1] "La vendimia durante la antigüedad, una tradición milenaria," 2021. [En línea]. Disponible en: https://museovinogalicia.xunta.gal/es/blog/la-vendimia-durante-la-antiguedad-una-tradicion-milenaria

[2] "Rendimiento(viticultura)," 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Rendimiento_(viticultura)

[3] "The jakarta persistence query language," 2024. [En línea]. Disponible en: https://jakarta.ee/learn/docs/jakartaee-tutorial/current/persist/persistence-querylanguage/persistence-querylanguage.html#_the_jakarta_persistence_query_language

[4] "Yaml ain't markup language," 2024. [En línea]. Disponible en: https://yaml.org/

[5] "Dart programming langage," 2024. [En línea]. Disponible en: https://dart.dev/

[6] 2024. [En línea]. Disponible en: https://mustache.github.io/

[7] "Spring framework," 2024. [En línea]. Disponible en: https://spring.io/projects/spring-framework

[8] "Openapi specification," 2024. [En línea]. Disponible en: https://swagger.io/specification/

[9] 2024. [En línea]. Disponible en: https://flutter.dev/

[10] 2024. [En línea]. Disponible en: https://taiga.io/

[11] 2024. [En línea]. Disponible en: https://github.com/plantuml/plantuml

[12] 2024. [En línea]. Disponible en: https://www.figma.com/

[13] 2024. [En línea]. Disponible en: https://h2database.com/

[14] 2024. [En línea]. Disponible en: https://www.mysql.com/

[15] 2024. [En línea]. Disponible en: https://www.jetbrains.com/idea/

[16] 2024. [En línea]. Disponible en: https://developer.android.com/

[17] 2024. [En línea]. Disponible en: https://github.com/

[18] 2024. [En línea]. Disponible en: https://git-scm.com/

[19] 2024. [En línea]. Disponible en: https://www.docker.com/

[20] 2024. [En línea]. Disponible en: https://www.atlassian.com/es/agile/project-management/scrumban

[21] 2024. [En línea]. Disponible en: https://swagger.io/resources/articles/adopting-an-api-first-approach/

[22] 2024. [En línea]. Disponible en: https://swagger.io/

[23] 2024. [En línea]. Disponible en: https://swagger.io/solutions/mocking-and-virtualization/

[24] 2020. [En línea]. Disponible en: https://openapi-generator.tech/

[25] 2020. [En línea]. Disponible en: https://spec.openapis.org/oas/v3.0.3

[26] 2020. [En línea]. Disponible en: https://springdoc.org/