Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención en computación

Curso académico: 2021-22

BiON TIME: Sistema web de marcaje online

Alumna:
Rocío Íñigo Lobo

Tutora:
Alejandra Martínez Monés

Tutor:
Héctor Ortega Arranz

Agradecimientos

Este trabajo no habría podido llevarse a cabo si no fuera gracias a las personas que me han apoyado durante su desarrollo, así como durante el transcurso de la carrera universitaria. En especial:

Mis compañeros, que con el tiempo han pasado a ser buenos amigos, por su apoyo en aquellos momentos de duda o frustración.

A mi pareja, por apoyarme durante el desarrollo de este trabajo y estar a mi lado cuando lo he necesitado.

A los profesores, en especial a mis tutores Alejandra Martínez y Héctor Ortega, por aportarme sus conocimientos y guiarme durante el desarrollo de este trabajo.

Y por último, a mis padres, por su apoyo durante estos años y por su paciencia conmigo en aquellas situaciones en que me sobrepasaba el estrés.

Resumen

El objetivo de este Trabajo de Fin de Grado es desarrollar una aplicación web que permita a los empleados de Biome Makers registrar su jornada laboral y gestionar sus eventos. Biome Makers es una empresa dedicada a la biotecnología, con sedes en diferentes partes del planeta y que, por tanto, debe gestionar diferentes zonas horarias y calendarios laborales. Una de las prioridades fue realizar una aplicación intuitiva, centrando el diseño en la usabilidad y en la interacción con el usuario.

La aplicación web está dividida en varias secciones, dos de ellas para la visualización y registro de la jornada laboral. La primera de ellas, ofrece junto con el registro de trabajo, un resumen de lo registrado en la fecha y semana actuales y de los eventos próximos. La segunda, se centra más en la modificación y visualización de los registros. Las siguientes dos secciones se centran en la visualización de eventos, tanto propios como de los compañeros y en la gestión de los propios. Por último, el usuario administrador contará con otras dos secciones destinadas a la gestión de usuarios y a la obtención de reportes sobre los datos de la jornada laboral de los mismos.

El proyecto se ha llevado a cabo utilizando los *frameworks* Laravel y Vue, junto con una base de datos en PostgreSQL. En cuanto a la gestión de proyecto, se optó por seguir metodologías ágiles, en concreto: *Scrum.*

Abstract

The aim of this Final Degree Project is to develop a web application that allows the Biome Makers employees to record their working hours and manage their events. Biome Makers is a biotechnology company with sites in different parts of the world, and therefore has to manage different time zones and work schedules. One of the priorities was to create an intuitive application, focusing the design on usability and interaction with the user.

The web application is divided into several sections, two of them oriented to the registration and visualization of working hours. The first of them, allows the recording of working hours and also shows a summary of daily and weekly records, and of upcoming events. The second one is focused on the visualization and modification of the records. The next two sections are focused on the visualization of events of the employees, both their own events and the events of the other colleagues. They also allow to manage each employee's own events. In addition, the admin user will have the options to manage users and obtain reports with users' workday data.

The project has been developed using Laravel and Vue frameworks, and a database based on PostgreSQL. In relation to project management, it has been used Scrum, an agile method.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este proyecto se propone la creación y diseño de una aplicación web que permita gestionar la jornada laboral de los empleados de Biome Makers de forma sencilla y acorde a las necesidades de la empresa que, tras haber contratado los servicios de otra aplicación, consideró necesario el desarrollo de una aplicación *ad-hoc* que supliese los problemas encontrados con la otra aplicación. Para comprender mejor la situación que llevó al desarrollo de este proyecto, en este capítulo se introducen los siguientes aspectos:

- El **contexto** en que se desarrolla el proyecto.
- Los **proyectos similares** que ya han sido desarrollados.
- Los **motivos** que han llevado a realizar este proyecto.
- Los **objetivos** del mismo.
- La **estructura** que presenta el resto del documento.

1.1. Contexto

Debido a la relativamente **nueva normativa** sobre medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo, publicada el 12 de marzo de 2019 [1] [2], muchas empresas se han visto obligadas a llevar un seguimiento más exhaustivo de la jornada de sus empleados. Concretamente, el decreto establece la necesidad de mantener un registro de la jornada laboral efectiva de los "trabajadores que no se hayan comprometido de forma expresa, individual o colectivamente, a realizar horas extraordinarias" con el fin de que la Inspección de Trabajo y Seguridad Social pueda asegurar y controlar el cumplimiento de los horarios.

El registro deberá incluir la hora de inicio y finalización de la jornada de cada trabajador. Además, los registros deberán conservarse durante cuatro años y estarán a disposición de los trabajadores, de sus representantes legales y de la Inspección de Trabajo y Seguridad Social.
Actualmente muchas empresas llevan a cabo este registro de forma manual, en cuadrículas impresas o bien utilizando hojas de Excel, gestionadas por el responsable de Recursos Humanos. Esta metodología, algo rudimentaria, lleva a necesitar más espacio físico para almacenarlo, posibles pérdidas de información y un trabajo extra para mantenerlo organizado. Sin embargo, hoy en día las soluciones tecnológicas han facilitado el trabajo a la hora de acumular cualquier tipo de información, haciendo lo anterior innecesario en muchas ocasiones.

Biome Makers[1] es una compañía global de biotecnología que elabora modelos sobre la funcionalidad del suelo con el objetivo de mejorar la productividad de los suelos cultivables. Fue creada en 2015 y ha desarrollado una tecnología patentada que integra la secuenciación del ADN y las tecnologías de computación ecológica para descifrar el microbioma del suelo. Se trata de la principal plataforma de análisis del suelo, que permite la integración de datos del suelo con otras tecnologías de agricultura de precisión. Está formada por un equipo multidisciplinar en régimen de trabajo remoto, con perfiles de horario muy diferentes y repartidos por todo el mundo. La heterogeneidad que supone ser una compañía con integrantes de diferentes partes del mundo y con roles muy distintos, supone un hecho importante y puede suponer una dificultad a la hora de realizar el registro horario. Por ello, se plantea como un caso de estudio interesante a la hora de diseñar una aplicación de apoyo a la gestión de horarios.

1.2. Motivación

En esta sección se pretende dar una explicación a los motivos por los que se ha desarrollado una aplicación web para permitir a los empleados de Biome Makers mantener un registro diario de su jornada laboral. Se describen en primer lugar algunas de las carencias con las que se encontró la empresa y en segundo lugar se presentan otras aplicaciones existentes para el control horario, describiendo por qué razones no cubren las necesidades planteadas por Biome Makers.

Biome Makers tiene muchos trabajadores en régimen de trabajo remoto a lo largo del mundo, desde Turquía, Francia y España, hasta Estados Unidos y Sudamérica, y con perfiles de horario de trabajo muy diferentes. Debido a esta heterogeneidad de localizaciones y horarios, se hace necesario desarrollar un sistema flexible a la hora de integrar jornadas continuadas, partidas, desiguales, o traslados. Se desea tener un software que se implante dentro de la VPN de la empresa a la que los empleados puedan acceder y de manera sencilla, registrar las horas que dedican a su trabajo, descanso, etc.

[1] `https://biomemakers.com/`

1.2.1. Carencias actuales

En un primer momento, la empresa contrató una aplicación dedicada a la gestión horaria, *BioTime*, concretamente la versión 7.0. [3]. Sin embargo, esta aplicación presentaba una serie de carencias, algunas de las cuales podrían incluso suponer un riesgo para la seguridad de la empresa en el momento de poner en marcha la aplicación. Esto, entre otras dificultades para usar la aplicación, llevó a pensar que la mejor solución, tanto a nivel económico como de seguridad, sería desarrollar una aplicación ***in-house***, de modo que solo se implementara aquella funcionalidad que resultase realmente necesaria a la empresa. De esta forma, se evitaría contratar los servicios de una aplicación que además de lo solicitado pudiera ofrecer funcionalidad que finalmente no fuera utilizada pero aun así fuese incluida en el precio.

Además, al tratarse de un proyecto desarrollado específicamente para esta empresa, esto permite personalizar el producto, atendiendo a los requisitos establecidos directamente por el cliente y con la posibilidad de realizar las pruebas con algunos de los que serán los usuarios finales de la aplicación.

1.2.2. Servicios similares

Existen algunas aplicaciones que implementan de forma similar la funcionalidad mínima del proyecto [4] [5] [6]. La empresa estudió las características de algunas de ellas pero ninguna ofrecía un servicio que cumpliera satisfactoriamente con los requisitos que se exigen ante la situación y particularidades de Biome Makers. En la empresa existen diferentes regímenes para los empleados, siendo necesario para algunos de ellos tener que fichar obligatoriamente en unas oficinas físicas particulares, mientras que otros podrían hacerlo en remoto. Además, la gran variedad de zonas horarias para el fichaje tanto remoto como físico sumaba más complejidad al dominio del problema. Debido a esto se decidió explorar algunas de las soluciones comerciales existentes.

Sin embargo, tampoco se logró dar con una solución que cumpliera con las expectativas. Entre las principales limitaciones que se encontró Biome Makers en la implantación de estas soluciones comerciales están los riesgos de seguridad que se originaban al levantar los servicios, la incompatibilidad del software con sistemas operativos que no fueran Windows, o la necesidad de disponer un recurso de altas prestaciones para alojar únicamente este servicio, entre otros. Finalmente, se decidió generar un software propio para poder llevar a cabo este servicio. De esta manera se podría ajustar el programa *in-house* a todos los requisitos necesarios que se describirán más adelante.

A pesar de que las soluciones existentes no hayan sido la elección final para Biome Makers, algunas de de ellas han servido como inspiración para este proyecto, y para tener una visión de las carencias que habría que mejorar. Entre las plataformas utilizadas como base para diseñar la aplicación web, se encuentran *absence* y *toggl*. Esta primera es una aplicación destinada al mismo objetivo que la que se pretende desarrollar, mientras que la segunda tiene un objetivo más general de gestión de proyectos.

Absence

Absence, [7], es una aplicación que permite visualizar un resumen de las ausencias del usuario, así como otros datos relacionados con el calendario en la pantalla principal. Otras opciones de visualización son el calendario propio y el de empresa, las ausencias y el registro horario. Además, permite gestionar al usuario la compensación de horas y el personal.

Aunque en lo que a funcionalidad respecta es una aplicación bastante completa, la interfaz puede resultar poco atractiva, no priorizando aquellas funcionalidades que se consideran principales, dando la misma importancia a la gestión de ausencias del usuario que al registro de su jornada laboral. De esta forma, el aprendizaje del usuario pueda verse ralentizado.

Por dichos motivos, se utilizará sobre todo como base para definir algunas funcionalidades y ampliar otras.

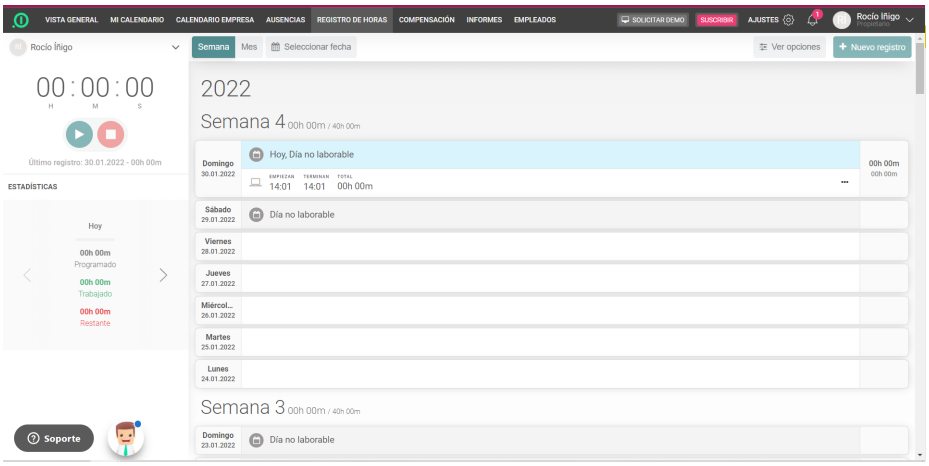


Figura 1.1: Muestra de la interfaz de Absence

Toggl

Toggl, [8], en contraposición a la primera, no es una aplicación destinada al fichaje de los empleados, sino al registro de actividades que puedan pertenecer a proyectos diferentes. Sin embargo, presenta una interfaz algo más atractiva, motivo por el que se elegirá como base para el diseño de este proyecto, ya que su funcionalidad es similar pero más genérica.

Permite visualizar los registros de inicio y finalización de actividades con distintos formatos, así como obtener reportes gráficos y gestionar los diferentes tipos de proyectos, clientes o equipos, además de utilizar etiquetas para agrupar las distintas actividades. Esto servirá de inspiración para diferenciar entre registros, así como proporcionar un resumen más visual de su jornada a los usuarios.

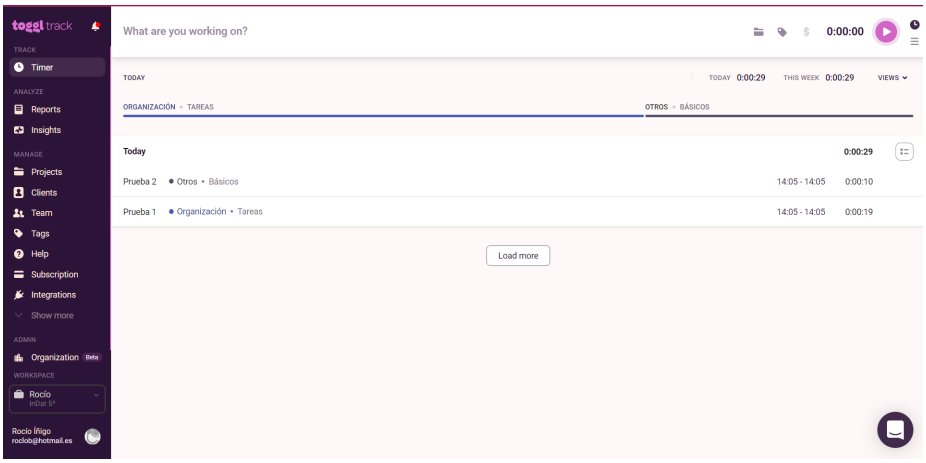


Figura 1.2: Muestra de la interfaz de Toggl

1.3. Objetivos del proyecto

En respuesta a las carencias arriba descritas se plantea el desarrollo de esta aplicación, que supone la creación de un proyecto ***in-house*** **basado en producto**, es decir, los clientes y desarrolladores están dentro de la misma organización y el objetivo es desarrollar un producto, concretamente una aplicación web que permita a los empleados de Biome Makers mantener un registro diario de su jornada laboral. Se trata además, de un **sistema de uso obligatorio**, puesto que, debido al nuevo reglamento sobre las horas de trabajo en España, será necesario su uso por parte de los empleados.

Por tanto, el objetivo principal de este trabajo es *desarrollar una aplicación web que permita a los empleados de Biome Makers registrar y visualizar sus jornadas laborales de un modo sencillo,*

cumpliendo así con la normativa vigente a este respecto. Además de esto, se permitirá la visualización de las vacaciones de los empleados, de forma individual y global, avisando de las posibles incompatibilidades con otros usuarios.

Se desea que la aplicación web contenga además algunas funciones de administrador para poder ver/estudiar y gestionar los datos de los usuarios.

1.4. Estructura del documento

La memoria se estructura en varios capítulos:

- El **Capítulo 2** presenta la *Planificación* del proyecto; es decir, las fases en que se dividió el desarrollo del proyecto, el seguimiento de las mismas, así como los posibles riesgos a contemplar y el presupuesto que suponga llevarlo a cabo.
- El **Capítulo 3** de *Análisis y Diseño* se dividirá en el análisis, donde se presentan los requisitos funcionales, no funcionales, de información y de usabilidad; y en el diseño, tanto del modelo de datos como de las interacciones.
- En el **Capítulo 4** se trata la *Implementación* del proyecto, siendo necesaria para la descripción del desarrollo una introducción a las diferentes tecnologías y lenguajes utilizados para llevar a cabo el mismo.
- Tras describir la implementación es necesario un capítulo dedicado a las *Pruebas*, el **Capítulo 5**, que permita comprobar la robustez de la aplicación desarrollada y conocer los casos contemplados en el comportamiento de la misma.
- Por último, se encuentra el **Capítulo 6** de *Conclusiones*, donde se analizará si realmente se han cumplido los objetivos, el desarrollo y pruebas realizados, así como el posible trabajo futuro a realizar.

Capítulo 2

Planificación

En este capítulo se van a explicar los siguientes aspectos:

- Qué es un **modelo de desarrollo *software***
- Los modelos de desarrollo *software* **más representativos**
- En qué consiste el **modelo de desarrollo software** elegido y por qué se ha optado por el mismo.
- Las **fases de desarrollo** del proyecto.
- El **seguimiento** de dichas fases de desarrollo.
- El **análisis de riesgos**.
- El **presupuesto** demandado por el proyecto.

2.1. Modelos de desarrollo software

A continuación, se explicarán los diferentes modelos de desarrollo *software* existentes y sus características básicas.

Para entender lo que es un modelo de desarrollo *software* o modelo del proceso, es necesario entender primero lo que es un proceso [9]. Obtener lo que en este caso sería un producto implica realizar una o varias actividades, es decir, realizar un proceso. La forma en que se organizan estas actividades es a lo que se llama **modelo del proceso** [10]. Una vez entendido esto, hoy en día, los modelos de desarrollo se pueden dividir en modelos estructurados o en modelos dinámicos [10]:

Entre los **modelos estructurados** podemos encontrar la aproximación orientada a objetos, según algunos autores. Estos métodos siguen unos pasos y unas reglas que llevan a obtener uno o varios productos, cada uno de ellos documentado cuidadosamente. Sin embargo, esto también implica un mayor coste debido al tiempo necesario para documentar el proceso frente a otras aproximaciones más intuitivas. Por ello, este gasto extra suele estar más justificado en proyectos que impliquen a muchos desarrolladores y usuarios debido a sus dimensiones, a diferencia del caso que se está presentando en esta memoria.

Los modelos estructurados más conocidos son el modelo en cascada y los modelos en V y en espiral. Estos últimos pueden considerarse variantes del primero.

2.1.1. Modelo en cascada

El **modelo en cascada** o ***one-shot*** [11] [12] [10] se caracteriza por presentar una secuencia de actividades o fases organizadas de forma lineal, que debe seguirse de la forma más estricta posible, permitiéndose en todo caso volver a la actividad anterior si se detectase algún problema que lo hiciese necesario.
Las fases en que se divide este modelo son las siguientes, señalándose las más básicas:

1. Estudio de factibilidad
2. Requisitos
3. **Análisis**
4. **Diseño**: Diseño del sistema + Diseño del programa
5. **Codificación**
6. **Pruebas**
7. **Ejecución/Verificación**

Las fases de estudio de factibilidad y requisitos se pueden incluir en la fase de análisis, junto con la planificación. Por su parte, las fases de codificación y pruebas unitarias pueden englobarse en una más general de **implementación**, separándose del resto de pruebas. Además, podría contemplarse una fase extra de **mantenimiento**.
En general, debe seguirse el orden de las fases y estas deben ejecutarse idealmente una sola vez.

Entre las **ventajas** del modelo en cascada, esto último puede considerarse en cierto modo el punto fuerte del modelo, ya que obliga a completar determinadas actividades antes de sus *deadlines*, facilitando así la completitud a tiempo del producto y una representación cronológica sencilla. Además, en aquellos proyectos en los que las fases son claramente diferenciables, permite

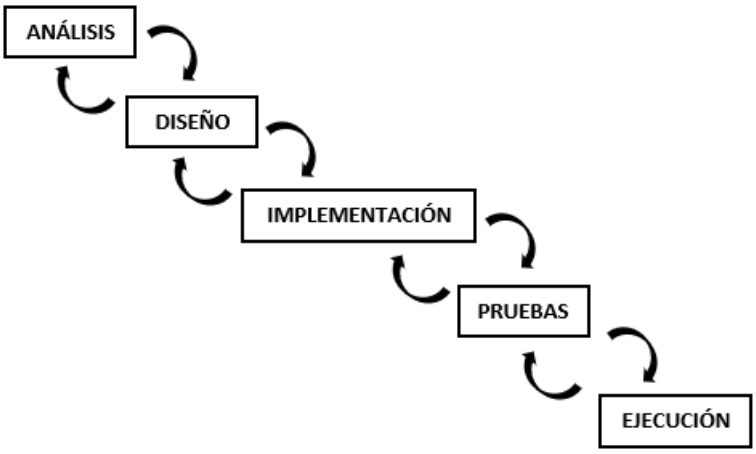


Figura 2.1: Modelo en cascada

mantener una organización clara y fácil de comprender. Otro de los beneficios de este modelo es la documentación detallada, lo que facilita comprender cada uno de los pasos que se ha seguido durante el proceso de desarrollo.

Por otro lado, también presenta algunas **desventajas**, ya que en la práctica las fases no suelen estar tan separadas unas de otras, sobre todo en los proyectos más complejos, caso en el que realmente podría estar justificado dedicar un alto coste a la documentación, o donde los desarrolladores no tienen demasiada experiencia, como es el caso. De forma general, la principal desventaja de este modelo es que no es realista, ya que no se adapta a las potenciales modificaciones que pueden resultar necesarias a lo largo del proceso. Esto es mucho más probable en proyectos de gran envergadura, ya que al prolongarse en el tiempo, deberán adaptarse a los cambios que puedan darse durante el mismo. Además, al no interactuar con el cliente hasta que no se ha finalizado la fase de implementación es posible que se detecten bastantes fallos que podrían haberse resuelto con anterioridad, evitando la necesidad de modificar un conjunto de elementos mucho mayor. En el caso de este proyecto, esto es algo que podría evitarse, dado que existe la posibilidad de mantener una comunicación mayor con el cliente.

2.1.2. Modelo en V

El **modelo en V** o **modelo en cuatro niveles** [13] [10] podría considerarse una variación del modelo en cascada que desdobla el testado en partes para comprobar la corrección y validar cada una de las actividades previas a la codificación.
Lo ideal sería que esta validación solo se realizase en caso de encontrar una discrepancia entre lo especificado en una actividad concreta y lo que realmente se ha implementado en la siguiente actividad.

Las cuatro fases de las que es posible hablar son: un estudio de factibilidad, la especificación de los requisitos, una fase de diseño y otra fase de codificación. Una vez se pone fin a la fase de codificación se dividen las pruebas o verificación del producto, de forma que permitan evaluar el resultado de cada una de las fases mencionadas antes, realizando esta evaluación solo en caso de que las pruebas sobre la fase siguiente no hayan sido satisfactorias.

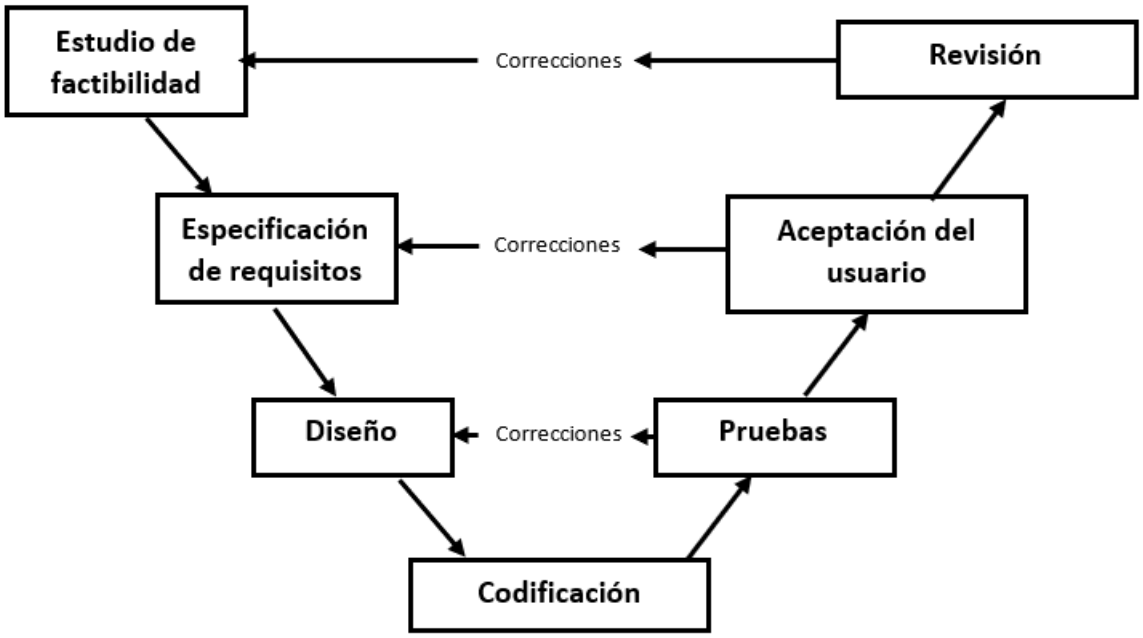


Figura 2.2: Modelo en V

Con la aparición de los modelos ágiles, se ha desarrollado una variante del modelo en V, que contempla la posible eliminación de algunas fases en proyectos que no sean lo suficientemente grandes como para que merezca la pena el coste que supone la realización de todas las fases en las que consiste el modelo básico. Además, en ese caso se prevé también la inclusión del cliente en fases más tempranas a la verificación del producto.

Entre las **ventajas** del modelo en V, se encuentra, como en el caso del modelo en cascada, la claridad y fácil comprensión que permite del proceso. Además, la división de las pruebas permite un mayor control de la calidad del producto resultante. Al igual que el modelo en cascada, también permite ajustarse de forma estricta a las *deadlines* del proyecto.

Las **desventajas** de este modelo, además de las ya indicadas en el modelo en cascada por ser ambos demasiado rígidos, incluyen una posible simplicidad excesiva, al dividir el proceso en cuatro fases. Algunas de estas desventajas podrían evitarse adaptando el modelo a las metodologías ágiles, como es el caso de la falta de interacción con el cliente.

2.1.3. Modelo en espiral

El **modelo en espiral** [14] [10] se puede considerar también una variante del modelo en cascada. Este modelo, dirigido principalmente a la gestión de riesgos, tiene en cuenta la posibilidad de que la realización de algunas actividades más en detalle lleve a reconsiderar las decisiones tomadas en base a actividades anteriores. Por ello, en cada etapa del proyecto se considera en un mayor grado de detalle las actividades del proceso que permiten obtener el producto. Es decir, no se considera un orden lineal sino una organización iterativa del proceso, repitiendo las fases en forma de espiral o cíclica. Esto ralentiza el avance relativamente, pero permite reducir los riesgos al aumentar el control sobre las fases.

En una primera fase se suelen asignar diferentes objetivos a cada una de las iteraciones, definir los costes o los diseños iniciales. En la siguiente fase se evalúan los posibles riesgos y se eligen las alternativas, en caso de haber varias opciones, menos arriesgadas para conseguir el objetivo. Una vez se haya finalizado el análisis de riesgos, se da comienzo al desarrollo, que se realiza por ciclos, cada uno de ellos con uno o varios objetivos por cumplir.

En cuanto a las **ventajas** de este modelo, se puede considerar la más importante que, a pesar de seguir siendo un modelo estructurado, se adapta con más facilidad a proyectos grandes y complejos, ya que da un papel importante al análisis y gestión de riesgo, maximizando el control de costes. Es, en general, un modelo más flexible que el resto de modelos estructurados. Su estructura cíclica facilita la resolución de conflictos entre diseño y requisitos al poder revisar las decisiones tomadas. También permite la integración de los usuarios en etapas más tempranas del proceso.

Entre las **desventajas** podría incluirse la posibilidad de acercarse a las etapas finales con partes importantes inacabadas, existiendo así riesgo de presentar errores, pese a las revisiones constantes. Además, las modificaciones durante una etapa siempre pueden suponer retrasos en la siguiente. Por otro lado, el esfuerzo necesario para obtener resultados satisfactorios es bastante alto, por lo que no es conveniente utilizarlo en proyectos pequeños con riesgos más manejables.

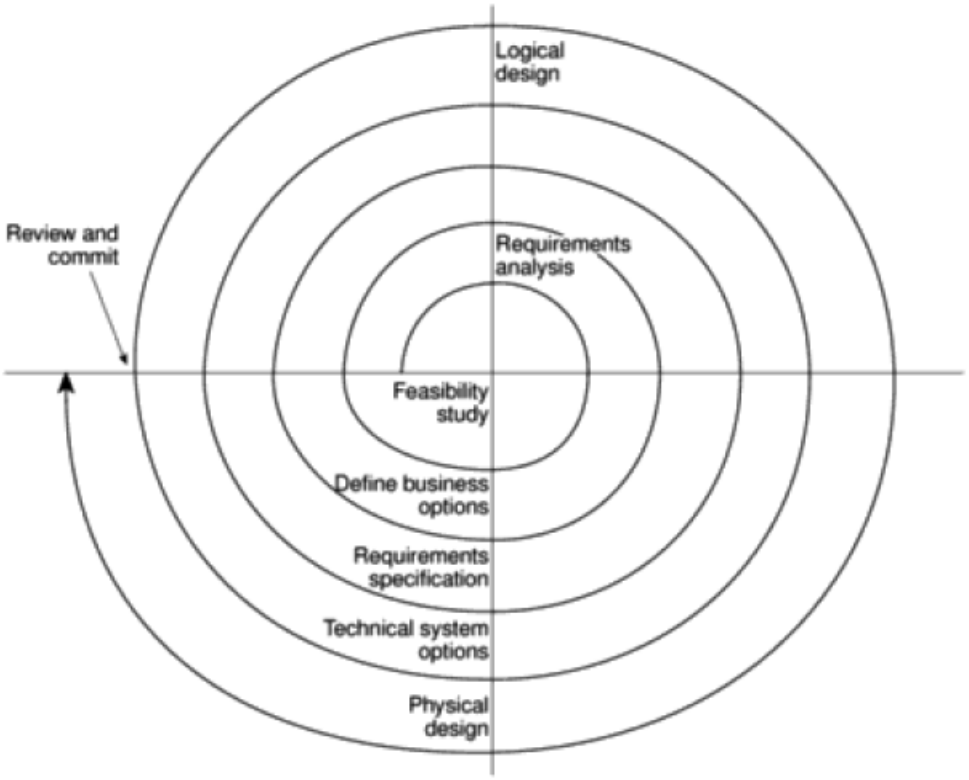


Figura 2.3: Modelo en espiral

2.1.4. Metodologías ágiles

Frente a las carencias de los modelos estructurados surgieron los **modelos dinámicos** o **metodologías ágiles**, basados en una forma de desarrollar iterativa e incremental, que permiten por su parte obtener resultados rápidamente y con un coste menor, centrándose en el tiempo de entrega en lugar de en la documentación.

La aproximación RAD (*Rapid Application Development*) [10] se centra en el uso de prototipos desarrollados rápidamente, pero no impide el uso de algunos elementos de las metodologías estructuradas. Incluye el uso de tácticas como talleres JAD (*Joint Application Development*), en los que los desarrolladores y usuarios trabajan juntos para asegurarse de que los requisitos del producto son los correctos. Otra práctica asociada a estos modelos es el *time-boxing*, donde el alcance del proyecto se restringe de forma estricta a una *deadline* previamente acordada y relativamente cerrada. Los requisitos que deben ser omitidos para cumplir con esta *deadline* se consideran en las siguientes *time-boxes*.

La diferencia principal con los modelos estructurados es que este tipo de metodologías contemplan la evolución de los requisitos y las decisiones tomadas con el tiempo y priorizan, como se

declaró en el *Manifiesto por el desarrollo ágil de software* [15], los **individuos e interacciones** sobre los procesos y herramientas, el ***software* funcional** sobre la documentación detallada, la **colaboración con el cliente** sobre la negociación contractual y la **respuesta ante el cambio** sobre seguir un plan.

En cada iteración, a diferencia del modelo en espiral, el modelo estructurado más cercano conceptualmente a las metodologías ágiles, se genera un entregable que, aunque no cuente con la funcionalidad completa del producto objetivo, sí funcione o no presente errores.
Algunos de los métodos ágiles más conocidos son las tecnologías *Crystal*, *Atern, DSDM*, *Extreme Programming, XP* o *Scrum*. A continuación se describe el *Scrum* por ser el método ágil más común y el elegido en este proyecto.

2.1.5. Metodologías ágiles: Scrum

Concretamente, el *Scrum* consiste en un marco de desarrollo ágil que surge a finales de los 80 y principios de los 90, antes de que se estableciera el Manifiesto Ágil, con el objetivo de desarrollar productos cada vez más complejos adaptándose a los cambios potenciales que pudiesen surgir [16].

A diferencia de los modelos clásicos como el de cascada, las metodologías ágiles [17] [18] y en concreto el *Scrum* contemplan la posibilidad de modificar las bases del proyecto conforme aumenta la experiencia y conocimientos de los integrantes del mismo. Esto es algo más realista dado lo que suele suceder en la práctica con los proyectos *software*, bien por cambios en los requisitos del proyecto, por caer en la cuenta durante el desarrollo de algo que inicialmente no se había contemplado, etc.

En el proceso característico de *Scrum* se busca seguir un conjunto de buenas prácticas para trabajo colaborativo, siguiendo una estrategia de desarrollo incremental, permitiendo cambios en la planificación y solapar las fases de desarrollo en lugar de seguir un orden estricto como se exige en los modelos estructurados.

Para entender bien esta metodología de trabajo es necesario conocer los elementos que la integran o **componentes** [19]:

Artefactos

El ciclo de vida del *Scrum* tiene una duración de normalmente una a cuatro semanas. Al final de cada *sprint* se presenta un producto potencialmente entregable o **incremento**, que deberá ir acercándose al producto final deseado. Este incremento es uno de los artefactos que forman parte del *Scrum*. Junto a este, otro artefacto a tener en cuenta es el ***Product Backlog***, una lista de requisitos o tareas ordenados por prioridad relativos al producto deseado, en forma de *user stories* o historias de usuario, que va evolucionando con el desarrollo del proyecto. La gestión del mismo es responsabilidad del *Product Owner*.

Las ***user stories*** podrían definirse como explicaciones generales realizadas de forma informal de una característica *software* desde la perspectiva del usuario final. Debe especificarse cuándo estaría completa dicha historia de usuario.

Una versión reducida del *Product Backlog* es el ***Sprint Backlog***, un subconjunto de dicha lista de requisitos que deben intentar cumplirse en el *sprint* correspondiente y escritos desde el punto de vista de los desarrolladores. Las estimaciones son actualizadas diariamente en lo que se conoce como *Daily Scrum*.

Scrum Team.

El equipo Scrum [20] [21] está formado por los siguientes integrantes: el *Scrum Master*, el *Product Owner* y el equpo de desarrollo o *Development Team*.

El ***Scrum Master*** es el encargado de asegurar que se sigue la metodología establecida y trabaja con el resto del equipo para resolver los posibles problemas que puedan surgir. En este proyecto, este rol corresponde a: Héctor Ortega, uno de los tutores del TFG.

El ***Product Owner*** representa al cliente y a los interesados en el producto final, los *stakeholders*. Se encarga de comprobar en cada *sprint* que el desarrollo del producto se está realizando conforme a lo deseado hasta el momento, así como de priorizar unos requisitos frente a otros de cara al siguiente *sprint*. Es decir, gestiona el *Product Backlog*. Además, ayuda a escribir las historias de usuario. Este papel corresponde a Merche López y Pedro Langa.

El **equipo de desarrollo** es el encargado de llevar a cabo las tareas establecidas en cada *sprint*. Suele estar formado por pocos miembros, entre 3 y 9 personas. En este caso, al tratarse de un proyecto no demasiado grande, los miembros del equipo serían tres: Rocío Íñigo, Eneas Marín y Javier Fresno.

Eventos.

El evento principal a la hora de hablar de *Scrum* es el *sprint*. Se trata de períodos de tiempo consecutivos, de entre una y cuatro semanas, en los que el producto va evolucionando de forma incremental. Se pueden definir cuatro tipos de eventos en función de este:

El ***Sprint Planning***: Es la primera actividad del *sprint*. En él, el cliente o *Product Owner* establece los requisitos priorizados del *sprint* correspondiente, es decir, el *Sprint Backlog*. No debería durar más de 8 horas.

Dentro del *sprint* se realiza a diario lo que se conoce como ***Daily Scrum***: Dura 15 minutos aproximadamente y en él se identifican los problemas y la situación de cada miembro del equipo desde la realización del *daily* anterior.

Al final del *sprint* se localiza el ***Sprint Review***, en el que se evalúa la situación actual del

producto, de forma que sea o no aceptado por el *Product Owner*. Se realiza un análisis general de lo ocurrido durante el *sprint*. Junto a este, tiene lugar el ***Sprint Retrospective***: Donde se genera *feedback* para solucionar los problemas que se hayan podido dar durante el proceso. Su duración es de una hora como máximo.

Durante cada *sprint*, los requisitos (es decir el *Sprint Backlog*), permanecen congelados, no pueden cambiar hasta el siguiente *sprint*.

Además, como ya se ha explicado, en *Scrum* se realizar entregas parciales del producto final periódicamente, priorizando cada entrega según el beneficio que aporten al cliente. Para priorizar estas entregas parciales se establecen los conceptos de ***stories***, ***epics*** e ***initiatives*** [22].

- Las ***stories*** o *user stories* son requisitos cortos, explicaciones genéricas e informales de una característica *software*, escritas desde la perspectiva de un usuario final. Se suelen poder completar en un *sprint* de una o dos semanas.
- Las ***epics*** son conjuntos de trabajo que pueden subdividirse en tareas más pequeñas conocidas como *stories*. Las *stories* que componen una *epic* están relacionadas entre sí.
- Las ***initiatives*** son conjuntos de *epics* con un objetivo común.

Entre las **ventajas** del *Scrum* se encuentra la posibilidad de presentar al cliente resultados anticipados, permitiendo así realizar correcciones antes de que las consecuencias dificulten cumplir con los plazos acordados, así como utilizar las funcionalidades del producto implementadas hasta el momento. Esto implica también reducir los riesgos y adaptarse a los cambios que puedan surgir. Al tomarse decisiones a tiempo real también se conoce con más detalle el tiempo medio de trabajo del equipo, lo que permite predecir cada vez con más precisión el tiempo que será necesario para completar el resto de funcionalidades.
Además, es recomendable en aquellos casos en que los requisitos están poco definidos.

2.2. Modelo elegido

Tras analizar las características de las metodologías ágiles y de los modelos estructurados, se ha llegado a las siguientes conclusiones:

- Los **modelos estructurados,** aunque cuentan con una documentación detallada, también suponen un **coste mucho mayor** que en este caso se ve innecesario, puesto que es más recomendable cuando se está trabajando en proyectos grandes con muchos desarrolladores, mientras que en este se ve involucrado un número pequeño de participantes.
- Por otro lado, la **falta de interacción con el cliente** en los modelos estructurados supone una desventaja que se puede evitar, dado que existe la posibilidad de mantener un contacto constante con el cliente al tratarse de un proyecto ***in-house***.

- Además, la falta de experiencia en este caso también **dificulta la diferenciación entre fases** de desarrollo del producto. De esta forma, el uso de modelos como el modelo en cascada o el modelo en V, donde esta diferenciación es estrictamente necesaria y las fases deben llevarse a cabo siguiendo un orden lineal, no es el más indicado.

- Cuando hay **cierta incertidumbre** sobre el desarrollo del proyecto, con la consecuente inestabilidad y riesgo que eso supone, suele ser de mayor utilidad una **aproximación incremental más flexible e iterativa**, lo que lleva a pensar que sería más adecuado el uso de metodologías ágiles o, en todo caso, el modelo en espiral. Esto facilitaría adaptarse a un proceso de aprendizaje, donde se desconocen las herramientas que se van a utilizar inicialmente, como ocurre en este caso.

- Aunque el **modelo en espiral** sí permite la resolución de conflictos entre diseño y requisitos al reevaluar las decisiones tomadas, **no exige ningún entregable** funcional distinto al objetivo principal, lo que podría suponer errores en las fases finales del proyecto, que se podrían evitar recurriendo al uso de metodologías ágiles. Además, el esfuerzo requerido con este modelo lo hace poco recomendable para proyectos pequeños como este.

- Por su parte, el desarrollo mediante metodologías ágiles promete **resultados más rápidos y flexibles** en un contexto más realista que acepta el hecho de que tanto clientes como desarrolladores puedan cambiar de opinión o cometer errores.
  Por otro lado, la presentación de un **entregable** cada cierto periodo de tiempo facilita recibir un *feedback* del cliente adecuado que consiga evitar errores en fases finales del proyecto.
  A esto se añade la posibilidad de **mejorar la precisión de las estimaciones** de tiempo conforme avanza el proceso, reduciendo los riesgos que implica la falta de experiencia sobre este aspecto.

- Por último, otra de las ventajas de elegir metodologías ágiles es entrar en contacto con una forma de trabajar cada vez más extendida entre las empresas dedicadas a este sector.

Por todo esto, se ha optado por utilizar metodologías ágiles y en concreto, *Scrum*, para llevar a cabo este proyecto.

2.3. Fases de desarrollo del proyecto

En esta sección se explican las fases en las que se dividió el proyecto desde el punto de vista de la planificación, siguiendo la metodología de *Scrum*. Para establecer dichas fases, primero fue necesario definir las *initiatives, epics* y *user stories* del proyecto. Más adelante, en la sección de análisis se plantean los requisitos de la aplicación en función de estas.

Para establecer estas, así como sus prioridades, los objetivos y la motivación del proyecto, nos planteamos una **reunión inicial** a principios de febrero de 2021, de la que se obtuvo el siguiente resultado:

2.3.1. *Initiatives, epics* y *user stories*: visión general

Como ya se ha explicado, en *Scrum* se definen una serie de características *software* mediante lo que se conoce como *user stories* que permiten priorizar las entregas parciales que se realizan en cada *sprint*. Éstas a su vez se agrupan en lo que se conoce como *epics*, que pueden conformar una o varias *initiatives* dependiendo de las dimensiones del proyecto.

En este proyecto se consideran tres ***initiatives*** ordenadas por prioridad:

- La **primera** y más importante destinada a obtener un producto donde el cliente pueda registrar su jornada laboral, diferenciarse según su papel en la empresa y obtener un resumen de sus datos.
- La **segunda** *initiative* considerada se centra en la asignación de vacaciones y tiempos de inactividad, así como la visualización de los mismos.
- La **tercera** y última, añade a las anteriores la posibilidad de notificar al cliente en determinados casos, tanto de incidentes con los registros como de incidentes con las vacaciones o tiempos de inactividad.

Siguiendo el orden indicado, se explica a continuación las *epics* correspondientes a cada *initiative* y sus *user stories*.

Primera *initiative*: Registros y roles

Las *epics* correspondientes a la **primera *initiative*** se centran respectivamente en los roles de usuario, los registros y los reportes o resúmenes de dichos registros:

***Epic* 1: Roles.** El **objetivo** de esta *epic* es que los usuarios estén clasificados según su rol dentro de la empresa, distinguiendo entre usuarios con o sin rol de administrador.

Las ***user stories*** correspondientes a esta *epic* son:

- **US1 - Clasificación de usuarios:** Como usuario quiero tener diferentes permisos en caso de ser o no administrador del sistema, así como conocer las coincidencias con otros usuarios que tengan mi misma función dentro de la empresa.

- **US2 - Login:** Como usuario quiero poder acceder a la aplicación mediante un nombre de usuario y una contraseña que me hayan sido asignadas y que podrá modificar el usuario administrador si así se le solicita.

- **US3 - Gestión de usuarios por el administrador:** Como administrador del sistema quiero poder modificar, añadir y eliminar el resto de usuarios.

***Epic* 2: Registros.** Los **objetivos** de esta *epic* están motivados porque, como usuario, deseo mantener un registro de mis horas de trabajo, estableciendo una diferenciación entre horas de trabajo y horas de descanso y pudiendo acceder a ellas en todo momento. Este registro debe tener la opción de realizarse en tiempo real así como la posibilidad de modificarse. Además, en caso de ser el administrador del sistema quiero poder visualizar los registros del resto de usuarios.

Las ***user stories*** correspondientes a esta *epic* son:

- **US1 - Visualización de registros:** Como usuario quiero poder visualizar los tiempos de entrada y salida de mi jornada laboral, diferenciando entre tiempo de descanso y tiempo de trabajo. Además, si soy un usuario con permisos de administrador podré ver los registros del resto de usuarios.

- **US2 - Creación de registros:** Como usuario deseo poder registrar mis horas de trabajo y descansos en el mismo momento en que esté trabajando o descansando.

- **US3 - Modificación de registros:** Como usuario deseo poder modificar mis horas de trabajo y descanso.

***Epic* 3: Reportes.** Otro de los **objetivos** más importantes y que se desglosan en esta *epic* es que, como administrador del sistema, debo tener la posiblidad de obtener un documento donde se muestren los registros de la jornada laboral de todos los usuarios. Además, como usuario deseo poder ver los eventos próximos y un resumen de las horas trabajadas.

Las ***user stories*** correspondientes a esta *epic* son:

- **US1 - Obtención de reporte:** Como administrador del sistema deseo poder obtener un reporte en formato .pdf o .csv con los datos obtenidos mediante el sistema, en el que se incluya información como el nombre de usuario, las horas trabajadas por el mismo y las horas de descanso.

- **US2 - Resumen actualizado:** Como usuario deseo ver un resumen gráfico de la situación actual de mis registros y/o eventos.

Segunda *initiative*: Vacaciones y tiempos de inactividad

En la **segunda *initiative*** las *epics* se centran en la asignación de vacaciones y gestión de los tiempos de inactividad. Concretamente solo se diferencia una *epic* en este caso:

***Epic* 1: Vacaciones y tiempos de inactividad.** El **objetivo** en este caso como usuarios es poder solicitar períodos de inactividad. Además, deben poder ver en determinados casos los períodos de inactividad de otros usuarios, pudiendo filtrar por roles, de forma que se facilite la organización de los usuarios dentro de la empresa a la hora de solicitar nuevos períodos de inactividad o vacaciones.

Las ***user stories*** correspondientes a esta *epic* son:

- **US1 - Solicitud de períodos de inactividad:** Como usuario deseo poder solicitar vacaciones o fechas en las que no asistiré a mi puesto de trabajo.
- **US2 - Visibilidad de períodos de inactividad:** Como usuario quiero poder ver las vacaciones o bajas del resto de usuarios, así como permitir o no que los míos sean visibles para el resto.
  Como administrador del sistema deseo visualizar las vacaciones o bajas de todos los usuarios, independientemente de sus permisos de visibilidad.

Tercera *initiative*: Notificaciones

La **tercera *initiative*** tiene como **objetivo** que se notifique al administrador del sistema de las modificaciones en los registro, así como a los usuarios de las coincidencias en los tiempos de inactividad con otros usuarios que tengan roles similares y de aquellos registros de actividad que puedan salirse de los establecido. En este caso, al igual que en la anterior *initiative* se establece solo una *epic* que tendrá por tanto el mismo objetivo que la propia *initiative*.

Las ***user stories*** correspondientes a esta *epic* son:

- **US1 - Modificación de registros:** Como administrador del sistema quiero ser notificado cuando uno de los usuarios modifique algún registro.
- **US2 - Solicitud de tiempos de inactividad:** Como administrador del sistema quiero ser notificado de las solicitudes de vacaciones del resto de usuarios.
- **US3 - Horas trabajadas:** Como usuario deseo que el sistema me avise cuando haya algún problema con mis horas trabajadas.

2.3.2. *Initiatives, epics* y *user stories*: asignación de prioridades

En primer lugar, se establece la prioridad de las *initiatives, epics* y *user stories*, así como los *sprints* en los que deberá completarse cada *epic* o *user story* dependiendo de sus dimensiones. Aunque las *initiatives* se han descrito anteriormente en orden de prioridad, a continuación se explica más detalladamente la prioridad dada a cada uno de estas:

Una vez definidas, se establece que las *user stories* **primordiales** son las correspondientes a la ***Initiative 1***, es decir, la que se centra en el registro y seguimiento de la jornada laboral de los trabajadores, así como de la identificación de los mismos. Dentro de esta, se priorizan las *user stories* correspondientes a los roles y los registros, en concreto las dos primeras *stories* de la *Epic 1*, que se centran en la identificación de los usuarios según sus roles, siendo menos prioritaria la gestión de los usuarios por el administrador, al existir la posibilidad de registrar a todos los usuarios dentro de la base de datos y no desde la aplicación.

Algo parecido ocurre con las *user stories* de la *Epic 2*, donde es primordial la visualización y creación de registros, ya que es necesario para el seguimiento y registro de la jornada laboral, quedando en segundo plano la modificación de registros.

Puesto que la *initiative* prioritaria es la relativa al registro de la jornada laboral, las *user stories* con **prioridad significativa** se centran en completar el objetivo de las *epics* correspondientes a dicha *initiative*. Así, se da mayor prioridad a la *user story* correspondiente a la obtención de reportes, ya que se estableció como un requisito obligatorio durante la reunión. El motivo de darle a esta *story* menos prioridad que a las cuatro anteriores, es que es necesario completar las anteriores para poder cumplir con el objetivo de esta.

Las siguientes *user stories* por orden de prioridad o de **prioridad moderada** son las correspondientes a la gestión de usuarios por parte del administrador (US 1.1.3), la modificación de registros (US 1.2.3) y la visualización de un resumen de los datos actualizado (US 1.3.2).

Las *user stories* de **prioridad leve** y **prioridad minoritaria** son las correspondientes a las otras dos *initiatives*, dado que la gestión de las vacaciones o tiempos de inactividad, así como las notificaciones no es algo básico dentro de la aplicación y se considera en todo momento algo que añadir a los requisitos establecidos por el cliente. Así, se considera de **prioridad leve** la *Initiative 2* y de **prioridad minoritaria** la *Initiative 3*, dado que las notificaciones serían una funcionalidad necesaria solo en algunas ocasiones.

A continuación se presenta la prioridad de cada una de las *user stories* de forma gráfica siguiendo un código de colores que se corresponde con la explicación dada anteriormente, donde el eje horizontal indica la urgencia con la que debe completarse cada *user story* y el eje vertical la simplicidad e importancia asociadas a dicha *user story*. De esta forma, el rojo se corresponde con las *user stories* prioritarias, el naranja con las de prioridad significativa, el ambar con las de prioridad moderada, el amarillo con las de prioridad leve y por último, el verde con las de prioridad minoritaria:

IMP \ URG	1	2	3	4	5
1	US.3.1.1 US.3.1.2 US.3.1.3				
2		US.2.1.1 US.2.1.2	US.1.2.3		
3			US.1.1.3	US.1.3.1	US.1.1.1
4		US.1.3.2		US.1.2.1 US.1.2.2	US.1.1.2

Tabla 2.1: Organización de *User Stories* por importancia y urgencia

Esta priorización de las *user stories* se realizó previamente utilizando *Trello*, concretamente el *plugin*: *Matrix for Trello*.

Antes, con el objetivo de tener clara la organización de las *user stories*, estas se crearon como tarjetas agrupadas por *epics* dentro de un mismo tablero como se muestra en la **Figura 2.4**. Dentro de cada sección, se encontraba además una tarjeta donde se describía la *epic*.

Figura 2.4: *User stories* en *Trello*

Las *user stories* estaban etiquetadas según un código de color, rojo para la *initiative* más prioritaria, naranja para la siguiente y amarillo para la última; y presentaban las siguientes secciones en su interior: Descripción, Requisitos asociados y Tareas asociadas, como se muestra en la **Figura 2.5**. Estos últimos correspondían a dos *checklists*, de forma que permitiesen ver desde fuera el progreso de cada tarjeta como se ve en la **Figura 2.6**, donde se observa además, cómo la tarjeta ya completada se sitúa al final del todo, cosa que puede automatizarse utilizando una de las funciones de *Trello*.

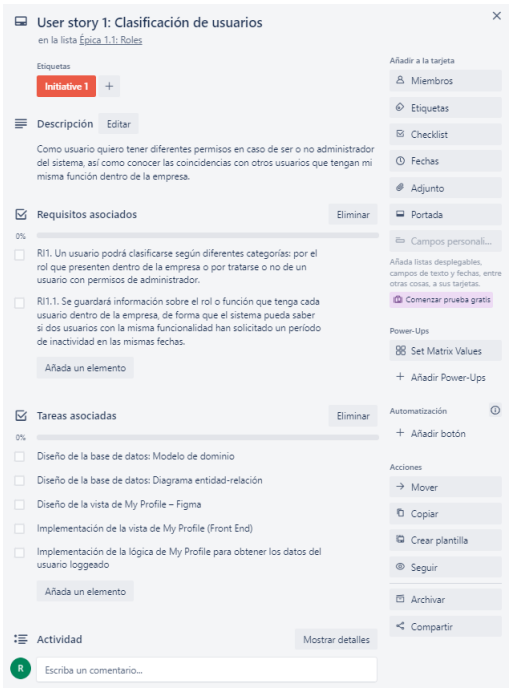


Figura 2.5: Tarjeta *user story*

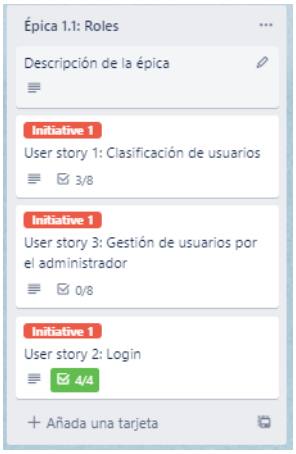


Figura 2.6: Grado de terminación *user stories*

2.3.3. *Sprints*

Siguiendo este orden de prioridades y en base también a la *epic* a la que corresponden las *user stories* se definen los cuatro *sprints* principales junto con la complejidad de cada tarea sobre 160

(las horas correspondientes a 4 semanas de trabajo con 5 días laborales y una jornada de 8 horas). Estos *sprints* podrán extenderse hasta a cuatro más, dos para solucionar los problemas generados durante la realización de los primeros y otros dos para la confección y revisión de la memoria. Se añade además un *sprint* inicial necesario para familiarizarse con las nuevas tecnologías que van a ser necesarias para el desarrollo de la aplicación y se incluye dentro de este el período utilizado para establecer y fijar los objetivos.

El ***Sprint Backlog*** correspondiente al **primer *sprint*** busca completar el *Epic* relacionado con las dos *user stories* de mayor urgencia, es decir el relativo a los roles de usuario. Así, las tareas correspondientes al ***SPRINT 1*** ordenadas por su prioridad dentro del *sprint* son:

- **Tareas prioritarias** relacionadas con la *User Story* 1.1.2: *Login*:
  - Diseño de la vista de *Login* y correspondientes interacciones (Figma).
  - Implementación de la vista de *Login* (*Front End*).
  - Implementación de la lógica de la vista de *Login* que comprueba si el par usuario-contraseña es correcto.
- **Tareas de prioridad moderada** relacionadas con la *User Story* 1.1.1: Clasificación de usuarios:
  - Diseño de la vista de *My Profile* y correspondientes interacciones (Figma).
  - Implementación de la vista de *My Profile* (*Front End*).
  - Implementación de la lógica de la vista de *My Profile* para mostrar los datos del usuario loggeado.
  - Diseño de la base de datos: Modelo de dominio.
  - Diseño de la base de datos: Diagrama Entidad-Relación.
- **Tareas de prioridad leve** relacionadas con la *User Story* 1.1.3: Gestión de usuarios por el administrador:
  - Diseño de la vista de *Management* y correspondientes interacciones (Figma).
  - Implementación de la vista de *Management* (*Front End*).
  - Diseño de la vista de *Add New User* y correspondientes interacciones (Figma).
  - Implementación de la vista *Add New User* (*Front End*).
  - Implementación de la lógica de la vista de *Management* para visualizar la lista de usuarios.
  - Implementación de la lógica de la vista de *Add New User* para añadir un usuario.

- Implementación de la lógica de la vista de *Management* para modificar/eliminar un usuario.

La estimación de complejidad de las tareas del primer *sprint* se muestra en la **Figura 2.7**.

TAREAS	Diseño e interacciones	Front End	Back End	Complejidad
US.1.1.2.	Login			2
US.1.1.2.		Login		6
US.1.1.2.			Login: user + psswd	6
PRIORITARIAS				14
US.1.1.1.	My profile			2
US.1.1.1.		My Profile		6
US.1.1.1.			My Profile: datos user	4
US.1.1.1.	Modelo dominio			16
US.1.1.1.	Diagrama E-R			16
MODERADAS				44
US.1.1.3.	Management			4
US.1.1.3.		Management		6
US.1.1.3.	Add New User			2
US.1.1.3.		Add New User		6
US.1.1.3.			Lista usuarios Management	3
US.1.1.3.			Add New User añadir usuario	20
US.1.1.3.			Management modificar/eliminar usuario	24
LEVES				65
SPRINT 1				123

Figura 2.7: *Sprint 1*

En cuanto al ***Sprint Backlog*** del **segundo *sprint***, este tiene como objetivo completar el resto de *epics* que se correspondan con la primera *initiative*. De acuerdo con esto, las tareas correspondientes ***SPRINT 2*** ordenadas por prioridad dentro del *sprint* son:

- **Tareas prioritarias** relacionadas con la *User Story* 1.2.1: Visualización de registros y la *User Story* 1.2.2: Creación de registros:
  - Diseño de la vista de *Timer List* y correspondientes interacciones (Figma). **Complejidad:**
  - Diseño de la vista de *Timer Daily* y correspondientes interacciones (Figma).
  - Diseño de la vista de *Timer Weekly* y correspondientes interacciones (Figma).
  - Implementación de la vista de *Timer List* (*Front End*).
  - Implementación de la vista de *Timer Daily* (*Front End*).
  - Implementación de la vista de *Timer Weekly* (*Front End*).
  - Implementación de la lógica de la vista de *Timer* para mostrar el resumen de horas.

- Implementación de la lógica de la vista de *Timer List* para mostrar la barra de resumen de horas en porcentaje.
- Implementación de la lógica de la vista de *Timer List* para mostrar la lista de registros.
- Implementación de la lógica de la vista de *Timer Daily* para mostrar el registro del día seleccionado.
- Implementación de la lógica de la vista de *TImer Weekly* para mostrar los registros de la semana seleccionada.
- Implementación de la lógica de *Timer* para crear un registro (*start/pause/stop*).

- **Tareas de prioridad moderada** relacionadas con la *User Story* 1.3.1: Obtención de reporte:
  - Diseño de la vista de *Reports* y correspondientes interacciones (Figma).
  - Implementación de la vista de *Reports* (*Front End*).
  - Implementación de la lógica de la vista de *Reports* para mostrar la tabla de registros.
  - Implementación de la lógica de la vista de *Reports* para filtrar los datos mostrados en la tabla.
  - Implementación de la lógica de la vista de *Reports* para seleccionar el formato de exportación del reporte.

- **Tareas de prioridad leve** relacionadas con la *User Story* 1.3.2: Resumen gráfico actualizado:
  - Diseño de la vista de *Summary* y correspondientes interacciones (Figma).
  - Implementación de la vista de *Summary* (*Front End*).
  - Implementación de la lógica de la vista de *Summary* para mostrar la tabla de eventos próximos.
  - Implementación de la lógica de la vista de *Summary* para mostrar el porcentaje de horas trabajadas.

- **Tareas de prioridad minoritaria** relacionadas con la *User Story* 1.2.3: Modificaicón de registros:
  - Implementación de la lógica de la vista de *Timer List* para modificar un registro.

La estimación de complejidad de las tareas del segundo *sprint* se muestra en la **Figura 2.8**.

TAREAS	Diseño e interacciones	Front End	Back End	Complejidad
US.1.2.1./2.	Timer List			4
US.1.2.1./2.	Timer Daily			4
US.1.2.1./2.	Timer Weekly			4
US.1.2.1./2.		Timer List		32
US.1.2.1./2.		Timer Daily		32
US.1.2.1./2.		Timer Weekly		32
US.1.2.1.			Mostrar resumen horas barra superior	8
US.1.2.1.			Resumen horas porcentaje Timer List	5
US.1.2.1.			Lista registros Timer List	20
US.1.2.1.			Registro día seleccionado Timer Daily	8
US.1.2.1.			Registros semana seleccionada Timer Weekly	6
US.1.2.2.			Creación registro start/stop/pause	32
PRIORITARIAS				187
US.1.3.1.	Reports			2
US.1.3.1.		Reports		8
US.1.3.1.			Tabla registros Reports	6
US.1.3.1.			Filtro tabla registros Reports	6
US.1.3.1.			Formato exportación registros Reports	12
MODERADAS				34
US.1.3.2.	Summary			4
US.1.3.2.		Summary		12
US.1.3.2.			Tabla eventos próximos Summary	4
US.1.3.2.			Porcentaje horas trabajadas Summary	8
LEVES				28
US.1.2.3.			Modificación registro Timer List	20
MINORITARIAS				20
SPRINT 2				269

Figura 2.8: *Sprint 2*

El ***Sprint Backlog*** correspondiente al **tercer *sprint*** se centra en completar la *Initiative 2* con una única *epic* y dos *user stories*, es decir, hacer funcional la gestión de los tiempos de inactividad. Las tareas del ***SPRINT 3*** ordenadas por prioridad dentro del mismo son las siguientes:

- **Tareas prioritarias** relacionadas con la *User Story* 2.1.2: Visualización de períodos de inactividad:
  - Diseño de la vista de *Company Calendar* y correspondientes interacciones (Figma).
  - Diseño de la vista de *My Calendar* y correspondientes interacciones (Figma).
  - Implementación de la vista de *Company Calendar* (*Front End*).
  - Implementación de la vista de *My Calendar* (*Front End*).
  - Implementación de la lógica de la vista de *Company Calendar* que permita ver las vacaciones de los usuarios.

- Implementación de la lógica de la vista de *Company Calendar* que permita filtrar por roles a los usuarios que se muestren.
- Implementación de la lógica de la vista de *Company Calendar* que permita elegir al usuario si sus vacaciones son o no visibles para el resto de usuarios.
- Implementación de la lógica de la vista de *Company Calendar* que indique la imposibilidad de ver determinados eventos.

- **Tareas de prioridad moderada** relacionadas con la *User Story* 2.1.1: Solicitud de períodos de inactividad:
  - Diseño de la vista de *My Calendar - Request* (Figma).
  - Implementación de la vista de *My Calendar - Request* (*Front End*).
  - Implementación de la lógica de la vista de *My Calendar* que permita solicitar vaciones o períodos de inactividad.

La estimación de complejidad de las tareas del tercer *sprint* se muestra en la **Figura 2.9**.

TAREAS	Diseño e interacciones	Front End	Back End	Complejidad
US.2.1.2.	Company Calendar			4
US.2.1.2.		Company Calendar		12
US.2.1.2.			Ver vacaciones usuarios en Company Calendar	7
US.2.1.2.			Filtro por roles en Company Calendar	8
US.2.1.2.			Elección de visibilidad vacaciones	4
US.2.1.2.			Notificación de falta de disponibilidad fechas	8
US.2.1.2.	My Calendar			3
US.2.1.2.		My Calendar		8
PRIORITARIAS				38
US.2.1.1.	My Calendar - Request Event			4
US.2.1.1.		My Calendar - Request Event		8
US.2.1.1.			Solicitar vacaciones en My Calendar - Request Event	24
MODERADAS				82
SPRINT 3				120

Figura 2.9: *Sprint 3*

Dado que el **cuarto *sprint*** era el último en orden de prioridad, puesto que estaba destinado a añadir la funcionalidad extra de las notificaciones y, a la vista de que la estimación de complejidad del proyecto era ya algo elevada, se decidió destinar el resto de *sprints* a corregir los posibles fallos que surgieran de los tres *sprints* anteriores y a la realización y corrección de la memoria. Habría que tener en cuenta además, un *sprint* inicial para entrar en contacto con las nuevas tecnologías y la toma de requisitos.

2.3.4. Diagrama de Gantt

En los diagramas de Gantt que se mostrarán a continuación pueden encontrarse las siguientes actividades desglosadas en diferentes tareas y dentro de diferentes *sprints*:

1. **Definición del proyecto**
    a) Toma de requisitos
    b) Revisión de requisitos
    c) Planificación del proyecto
2. **Diseño**
    a) Diseño UI|UX - Bocetos de las vistas
    b) Diseño de datos
    c) Diseño de la arquitectura
3. **Toma de contacto con las herramientas**
    a) Laravel
    b) Vue
4. **Desarrollo**
    a) Front-End
    b) Back-End
5. **Testeo y corrección de errores**

De todas estas actividades, algunas forman parte del **camino crítico**, es decir, aquel formado por las actividades que, en caso de producirse un retraso en la realización de las mismas, provocarían un retraso en el proyecto completo [23].

En este proyecto, las actividades de las que depende la implementación de la aplicación web son: la toma de requisitos, ya que es necesario conocer cuál es el producto deseado para comenzar su desarrollo y la toma de contacto con las herramientas necesarias para el desarrollo. Además, dentro del propio desarrollo habrá codependencias entre la implementación de la vista y el desarrollo *Back End*, así como entre la implementación de la vista y el diseño de la misma, aunque estas dependencias serán más leves y no comprometerán el desarrollo al completo.

En primer lugar, se muestra, en la **Figura 2.10**, una visión general del diagrama de Gantt, separado en 5 *sprints*, un *sprint* inicial para la toma de contacto en negro, dos *sprints* de color rojo dedicados a la primera *initiative*, un tercero de color naranja dedicado a la segunda *initiative* y por último, un *sprint* de color morado en el que se corregirán fallos y se realizarán las pruebas. En amarillo se muestra la posibilidad de añadir funcionalidad extra al finalizar este último *sprint*.

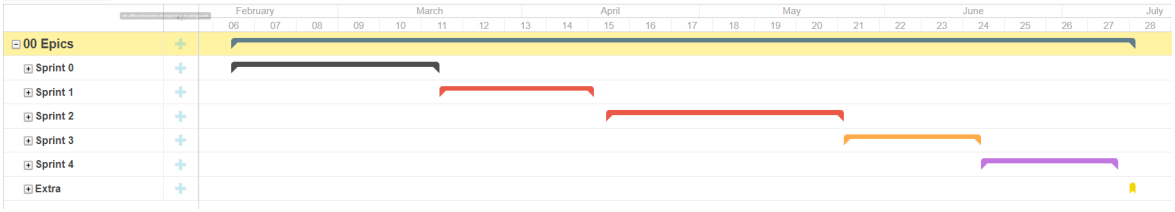


Figura 2.10: Diagrama de Gantt general

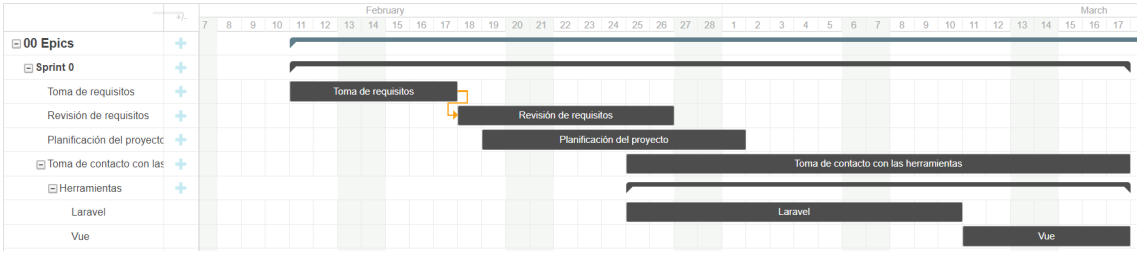


Figura 2.11: Diagrama de Gantt *Sprint* 0

En las **Figuras 2.12, 2.13 y 2.14** se muestra el despliegue del *Sprint 1* en el diagrama de Gantt:

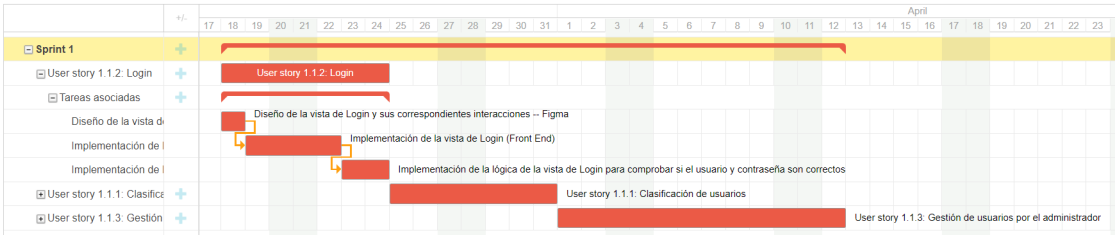


Figura 2.12: Diagrama de Gantt *Sprint* 1 US.1.1.2.

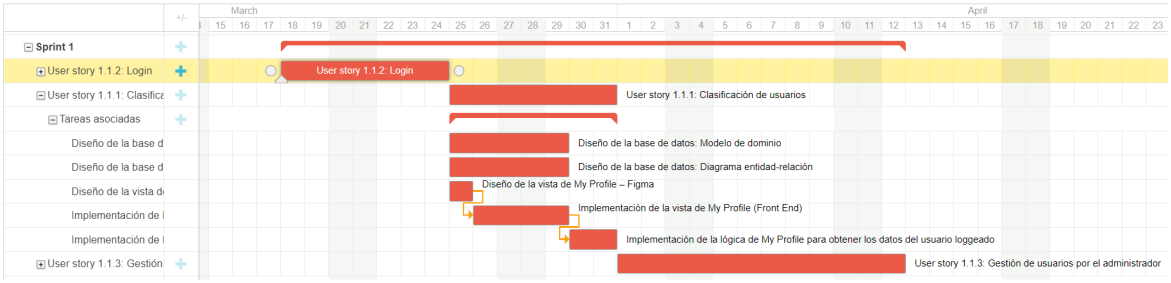


Figura 2.13: Diagrama de Gantt *Sprint* 1 US.1.1.1.

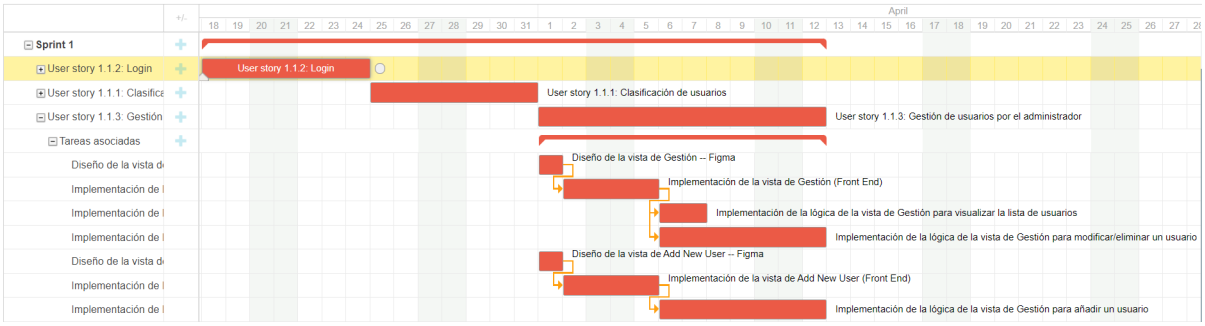


Figura 2.14: Diagrama de Gantt *Sprint* 1 US.1.1.3.

En las **Figuras 2.15, 2.16, 2.17, 2.18 y 2.19** se muestra el despliegue del *Sprint 2* en el diagrama de Gantt:

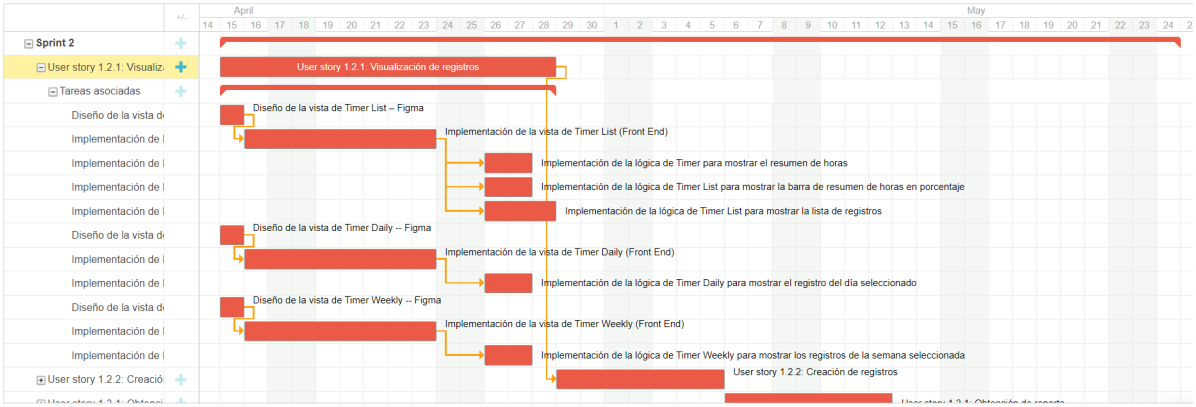


Figura 2.15: Diagrama de Gantt *Sprint* 2 US.1.2.1.

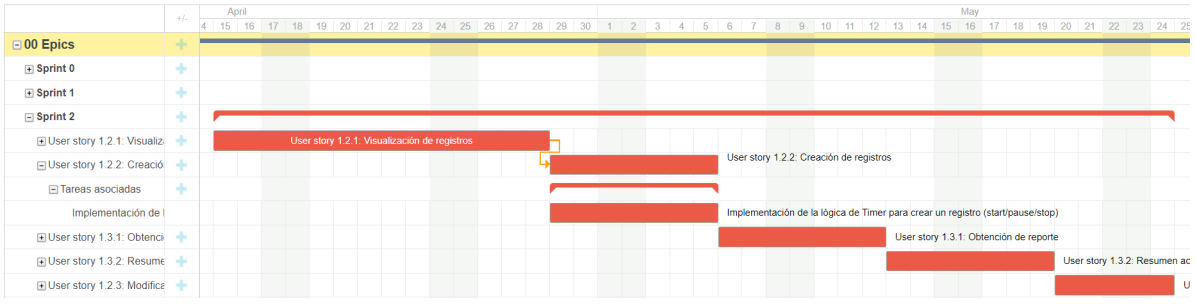


Figura 2.16: Diagrama de Gantt *Sprint* 2 US.1.2.2.

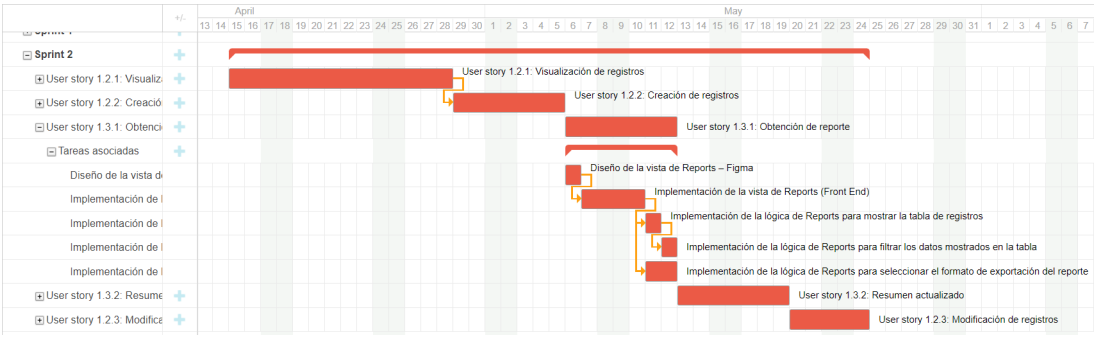


Figura 2.17: Diagrama de Gantt *Sprint* 2 US.1.3.1.

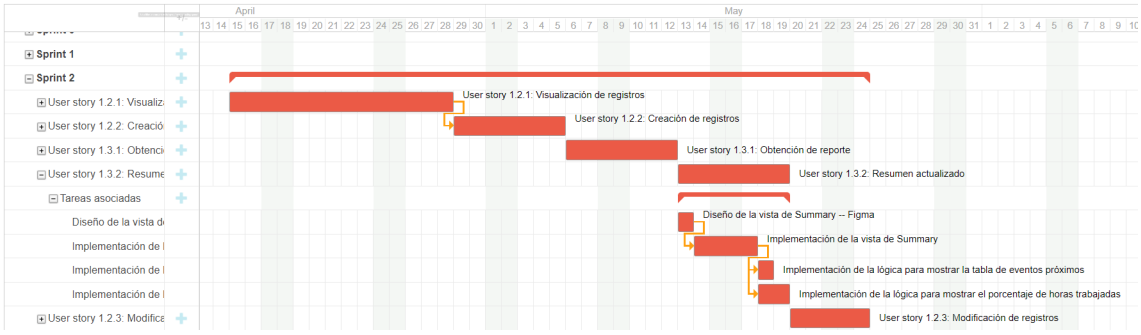


Figura 2.18: Diagrama de Gantt *Sprint* 2 US.1.3.2.

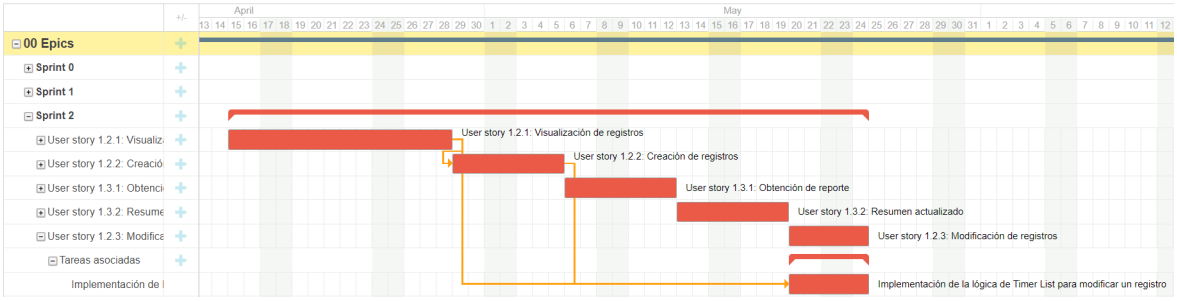


Figura 2.19: Diagrama de Gantt *Sprint* 2 US.1.2.3.

En la **Figura 2.20** se muestra el despliegue del *Sprint 3* en el diagrama de Gantt:

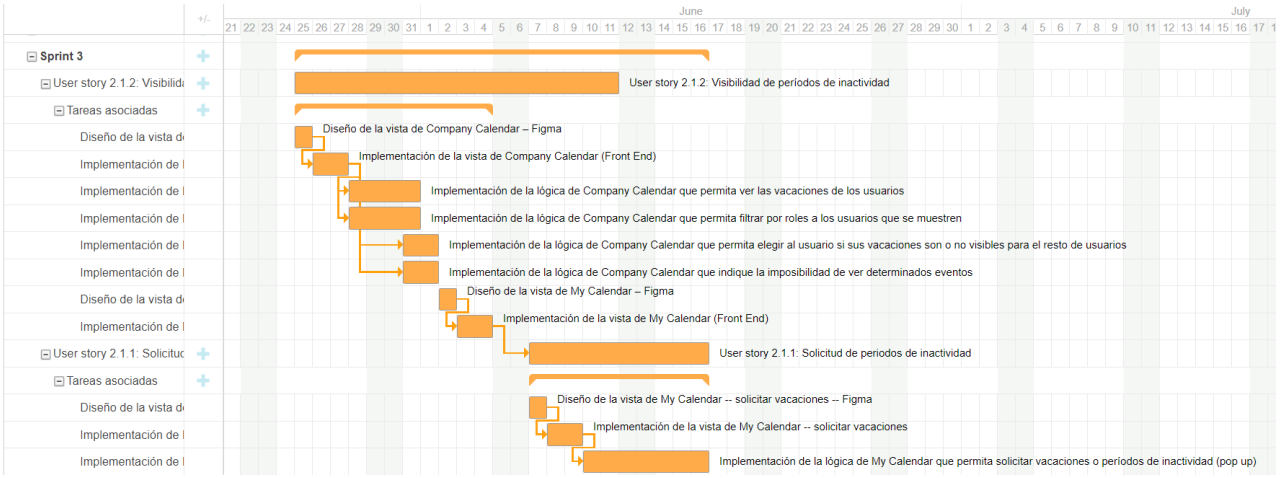


Figura 2.20: Diagrama de Gantt *Sprint* 3

En la **Figura 2.21** se muestra el despliegue del *Sprint 4* en el diagrama de Gantt:

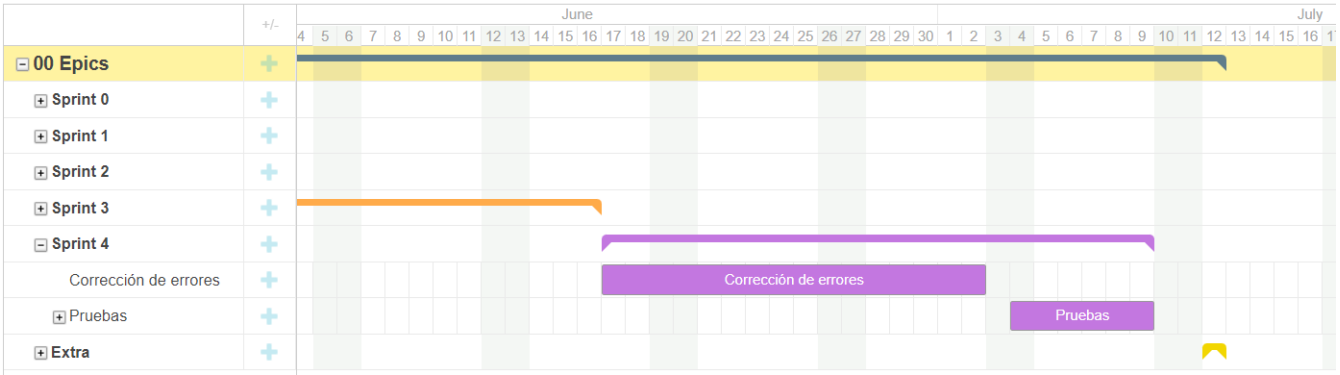


Figura 2.21: Diagrama de Gantt *Sprint* 4

2.4. Seguimiento

Esta sección pretende mostrar una retrospectiva respecto a las fases de la sección anterior, comprobando si realmente se han seguido y en qué medida el desarrollo del proyecto se ha desviado del plan inicial.

Para asegurar el avance del proyecto se realizaron reuniones periódicas. Durante los primeros meses estas reuniones se llevaron a cabo de forma presencial con el objetivo de solucionar problemas que pudiese generar la falta de conocimientos sobre las nuevas herramientas. Una vez la alumna se hubiese familiarizado con las mismas, estas reuniones pasarían a realizarse *online* y con una frecuencia más flexible (normalmente se realizaría una reunión semanal o quincenal, salvo

excepciones).

En lo que respecta al seguimiento del ***Sprint 0*** o inicial, la realización simultánea de otro Trabajo de Fin de Grado, al tiempo que cursaba una asignatura del doble grado, hizo que la adaptación a lo planificado fuese más lenta y prolongó la toma de contacto con las nuevas herramientas, de forma que fue necesario ampliar este primer *sprint* hasta mediados de abril, conllevando, al formar este parte del camino crítico, el consiguiente retraso del resto de *sprints*.

Además, la estimación del tiempo que iban a ocupar los *sprints* fue, en general, bastante optimista, teniendo en cuenta que, como se ha indicado, el proyecto se realizaría al mismo tiempo que otro Trabajo de Fin de Grado y una asignatura y, que además, muchas de las tecnologías necesarias para llevarlo a cabo eran totalmente nuevas para la alumna.

El estar poco familiarizada con las herramientas supuso un retraso importante al intentar poner en funcionamiento el proyecto *Laravel*, lo que causó ciertas dificultades con la instalación de algunos paquetes, que retrasaron el inicio del desarrollo. En cuanto a compatibilizar el desarrollo de ambos trabajos, esto no se tuvo en cuenta a la hora de establecer la planificación en el diagrama de Gantt. En este, se daba por hecho que la alumna podría dedicar al proyecto al menos 8 horas de trabajo todos los días. Sin embargo, dado el contexto, esto se hizo imposible, retrasando por tanto la planificación de todo el trabajo. A estos retrasos hay que añadir un objetivo quizá demasiado ambicioso que ya superaba por algunas horas las que debería suponer el desarrollo de un Trabajo de Fin de Grado en la primera estimación y otros imprevistos, como la contracción del Covid-19 a principios de julio.

Por otro lado, la alumna fue contratada para trabajar en una consultora a partir de noviembre, lo que supuso una reducción de las horas que podría dedicar a lo largo del día al proyecto.

Finalmente, todos estos contratiempos resultaron en las siguientes modificaciones del diagrama de Gantt, véanse **Figuras 2.22, 2.23, 2.25, ??, 2.26, 2.27, 2.28, 2.29 y 2.30**.

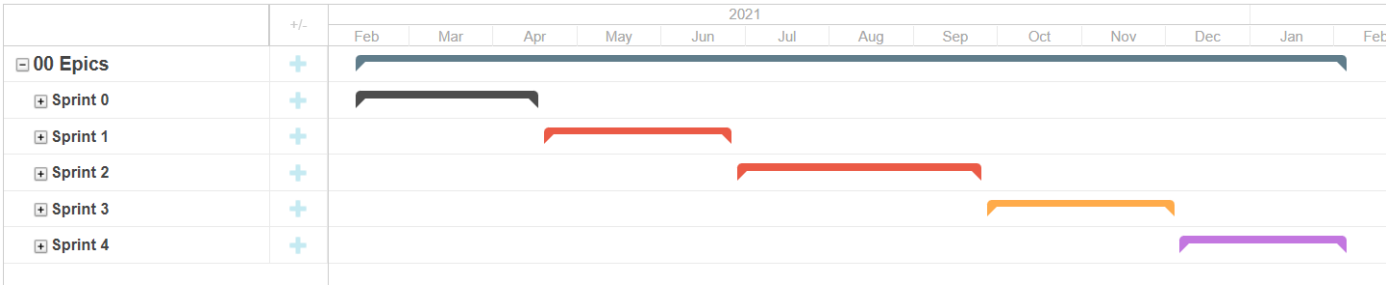


Figura 2.22: Diagrama de Gantt general

Figura 2.23: Diagrama de Gantt *Sprint* 0

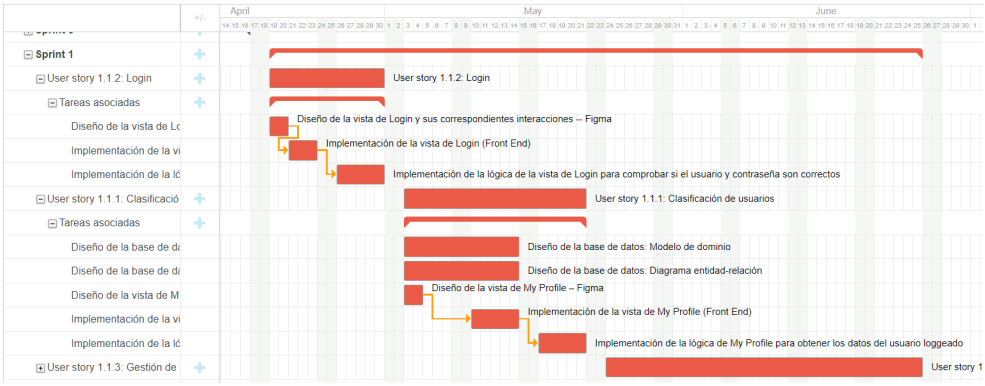


Figura 2.24: Diagrama de Gantt *Sprint* 1 US.1.1.2. + US.1.1.1.

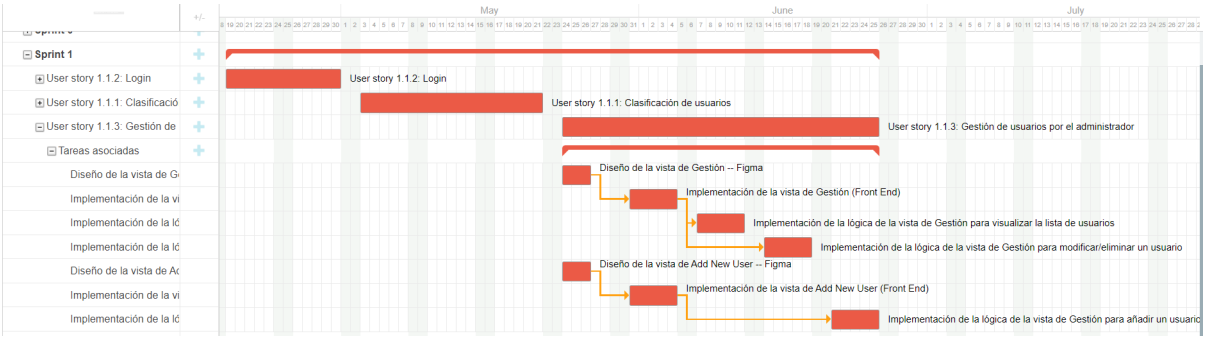


Figura 2.25: Diagrama de Gantt *Sprint* 1 US.1.1.3.

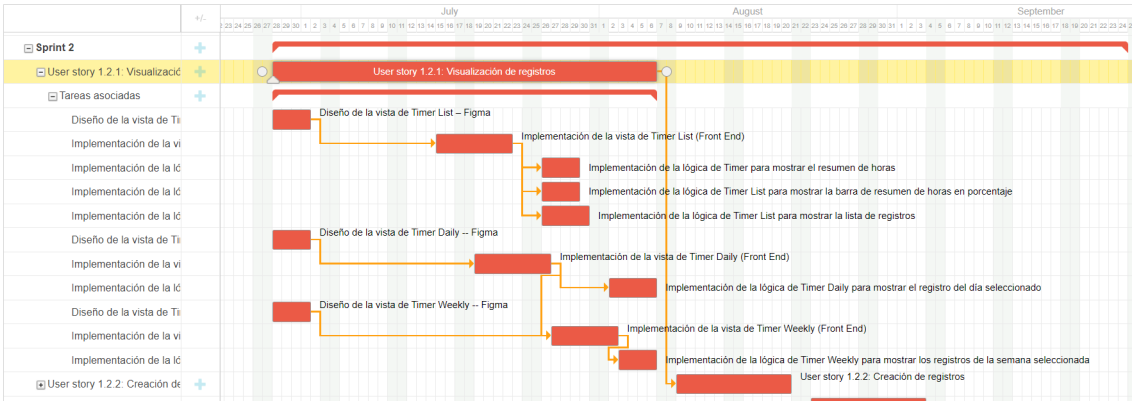


Figura 2.26: Diagrama de Gantt *Sprint* 2 US.1.2.1.

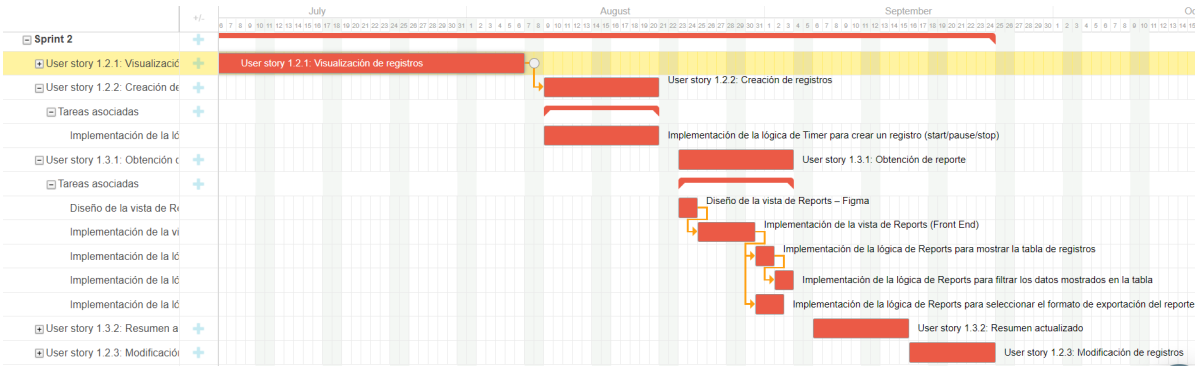


Figura 2.27: Diagrama de Gantt *Sprint* 2 US.1.2.2. + US.1.3.1.

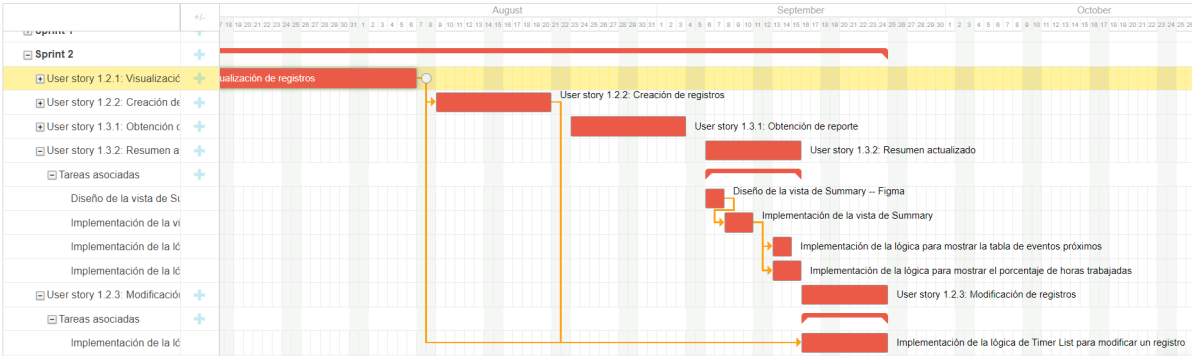


Figura 2.28: Diagrama de Gantt *Sprint* 2 US.1.3.2. + US.1.2.3.

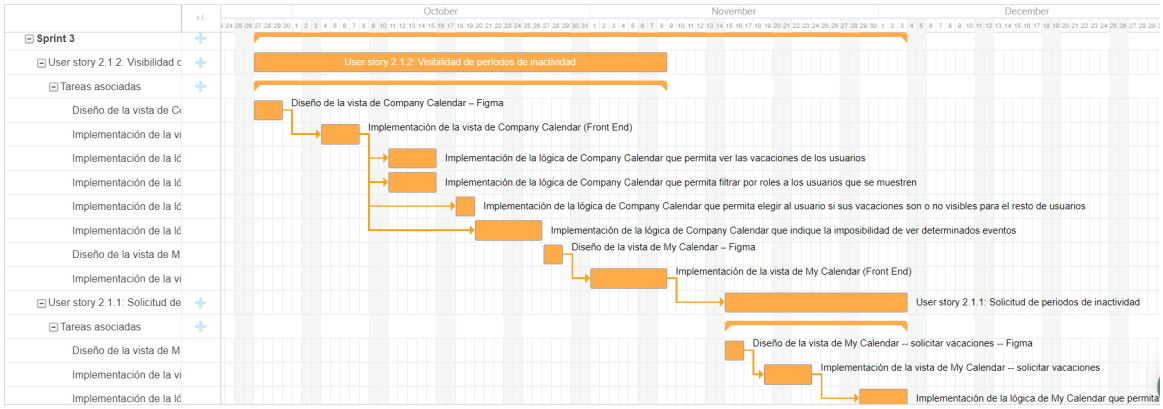


Figura 2.29: Diagrama de Gantt *Sprint* 3

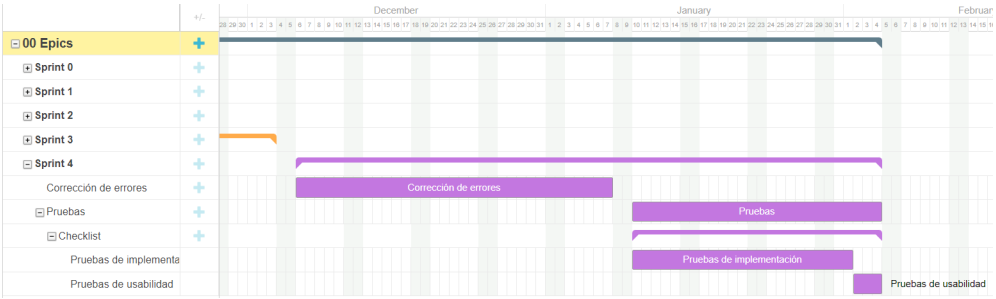


Figura 2.30: Diagrama de Gantt *Sprint* 4

En lo que respecta a las horas dedicadas al **cuarto *sprint***, estas se calcularían a partir de las horas estimadas en el resto de *sprints*, junto con el tiempo que se considerase necesario para realizar las pruebas.

En concreto, se multiplicarían las horas estimadas inicialmente por el porcentaje de tarea que se considerase completado, obteniendo así las horas ya dedicadas a esta tarea (las horas definitivas) como se muestra en las **Figuras 2.31, 2.32 y 2.33**.

TAREAS	Diseño e interacciones	Front End	Back End	Complejidad	Progreso	Total definitivo	Descripción
	Login			2	1	2	Diseño a papel y posteriormente en Figma + interacciones
		Login		6	1	6	Componente Vue + primera vista en Laravel + Vue (familiarizacion con componentes)
			Login: user + psswd	6	0,5	3	
PRIORITARIAS				14		11	
	My profile			2	1	2	Diseño a papel y posteriormente en Figma + interacciones
		My Profile		6	0,8	4,8	Componente Vue + modificaciones de los atributos mostrados y foto de perfil
			My Profile: datos user	4	1	4	Extraer datos del usuario de BBDD y mostrarlos en la vista
	Modelo dominio			16	0,5	8	Diseño y posteriores modificaciones / revisiones
	Diagrama E-R			16	0,5	8	Diseño y posteriores modificaciones / revisiones
MODERADAS				44		26,8	
	Management			4	1	4	Diseño a papel y posteriormente en Figma + interacciones
		Management		6	1	6	Componente Vue + filtro + vista modificación usuario
	Add New User			2	1	2	Diseño a papel y posteriormente en Figma + interacciones
		Add New User		6	1	6	Componente Vue: formulario + checklists + foto de perfil
			Lista usuarios Management	3	1	3	Lógica para mostrar usuarios en tabla Managemente
			Add New User añadir usuario	20	0,75	15	Problemas con checklists funciones y con foto de perfil
			Management modificar/eliminar usuario	24	0	0	Problemas con interacción base de datos
LEVES				65		36	
SPRINT 1				123		73,8	

Figura 2.31: Sprint 1: Horas calculadas + definitivas

TAREAS	Diseño e interaccion	Front End	Back End	Complejidad	Realizada	Total definitiv	Descripción
	Timer List			4	1	4	Diseño a papel + figma + interacciones
	Timer Daily			4	1	4	Diseño a papel + figma + interacciones
	Timer Weekly			4	1	4	Diseño a papel + figma + interacciones
		Timer List		32	1	32	Componentes Vue: TimerButton + Barra porcentajes + Registros
		Timer Daily		32	0,75	24	Componentes Vue: Horario día + interacción para pasar de días
		Timer Weekly		32	0,75	24	Compoenntes Vue: Horario semanal + interacción para pasar de semanas
			Mostrar resumen horas barra superior	8	1	8	Extraer suma de horas total / extra / planificadas
			Resumen horas porcentaje Timer List	5	1	5	Barra tamaño modifcable por porcentajes horas
			Lista registros Timer List	20	1	20	Actualización en tiempo real + calculo tiempo entre regsitros
			Registro día seleccionado Timer Daily	8	0,5	4	
			Registros semana seleccionada Timer Weekl	6	0,5	3	
			Creación registro start/stop/pause	32	0,5	16	Gestión de posts a BBDD y actualizacion del contador
PRIORITARIAS				187		148	
	Reports			2	1	2	Diseño a papel + figma + interacciones
		Reports		8	1	8	Filtros + tabla de registros + exportar pdf / csv
			Tabla registros Reports	6	1	6	Mostrar los registros en la tabla
			Filtro tabla registros Reports	6	1	6	
			Formato exportación registros Reports	12	1	12	Problemas con encontrar función de exportacion en Vue
MODERADAS				34		34	
	Summary			4	1	4	Diseño a papel + figma + interacciones
		Summary		12	1	12	Gráficos de círculo + tabla de eventos
			Tabla eventos próximos Summary	4	1	4	Extraer información eventos
			Porcentaje horas trabajadas Summary	8	1	8	Extraer porcentaje de horas trabajadas por dia y por semana
LEVES				28		28	
			Modificación registro Timer List	20	0	0	Estimado / Sin hacer
MINORITARIAS				20		0	
SPRINT 2				269		210	

Figura 2.32: Sprint 2: Horas calculadas + definitivas

TAREAS	Diseño e interacciones	Front End	Back End	Complejid	Pospuesta	Definitivas	Descripción
	Company Calendar			4	1	4	Diseño a papel + interacciones figma
		Company Calendar		12	1	12	Tabla para mostrar vacaciones + filtro por roles
			Ver vacaciones usuarios en Company Calendar	7	0,5	3,5	Filtro por tipo + roles + visibilidad
			Filtro por roles en Company Calendar	8	1	8	Filtro por rol similar al usuario
			Elección de visibilidad vacaciones	4	0	0	Check en My Calendar
			Notificación de falta de disponibilidad fechas	8	0	0	Mostrar aviso cuando se soliciten vacaciones
	My Calendar			3	1	3	Diseño a papel + interacciones figma
		My Calendar		8	1	8	Componentes: calendario + barra superior + botón
PRIORITARIAS				38		38,5	
	My Calendar - Request Event			4	1	4	Diseño a papel + interacciones figma
		My Calendar - Request Event		8	1	8	Componentes formulario
			Solicitar vacaciones en My Calendar - Request Event	24	1	24	Post a BBDD
MODERADAS				82		36	
SPRINT 3				120		74,5	

Figura 2.33: Sprint 3: Horas calculadas + definitivas

En el *Sprint 4* se añadirían las tareas que no estuviesen completas con la diferencia entre las horas estimadas y las horas ya dedicadas a las mismas, como se puede ver en la **Figura 2.34**.

TAREAS	Diseño e interacciones	Front End	Back End	Complejid
			Login: user + psswd	3
	Modelo dominio			8
	Diagrama E-R			8
			Management modificar/eliminar usuario	24
		Timer Daily		8
		Timer Weekly		8
			Registro día seleccionado Timer Daily	4
			Registros semana seleccionada Timer Weekly	3
			Creación registro start/stop/pause	16
			Modificación registro Timer List	20
			Ver vacaciones usuarios en Company Calendar	3,5
			Elección de visibilidad vacaciones	4
SPRINT 4				109,5

Figura 2.34: Sprint 4: Horas calculadas + definitivas

Como puede verse, los diferentes obstáculos encontrados a lo largo del desarrollo del proyecto y que supusieron no seguir la planificación inicial, ya de por sí demasiado optimista con el tiempo necesario para realizar algunas tareas, acabaron desembocando en la necesidad de continuar con el proyecto en un curso diferente al año en que se había iniciado el mismo.

2.5. Análisis de riesgos

En esta sección se presenta la planificación para la gestión de los potenciales riesgos del proyecto detectados al inicio del mismo.

Se conoce como riesgo [24] [10] todo aquel posible problema o circunstancia futura, no actual, que pueda darse en el proyecto por una determinada causa y que tenga un efecto negativo sobre los objetivos del mismo. Su probabilidad de incidencia es desconocida y diferente dependiendo del riesgo del que se trate.

El Plan de Gestión de Riesgos comprende cuatro pasos a seguir [25]:

1. Identificación del riesgo.
2. Análisis y priorización de los riesgos.
3. Planificación del riesgo.
4. Monitorización del riesgo.

En la matriz de la **Figura 2.35** se muestran los riesgos identificados, así como la priorización de los mismos. Para el cálculo de dicha priorización, se utilizará el valor conocido como RE o exposición al riesgo, que se obtiene como producto de la importancia o daño potencial y de la probabilidad de ocurrencia. Para asignar un valor a la importancia del riesgo, esta representará las semanas en que se podría retrasar el proyecto si se diese dicho riesgo. Es decir, si por ejemplo la toma de requisitos es incorrecta, esto supondrá una pérdida de 2 semanas del proyecto, una para volver a tomar los requisitos y otra para solucionar los posibles problemas que ese cambio pueda suponer para el desarrollo. En cuanto a la probabilidad, se medirá del 0 al 10, siendo 10 un suceso seguro y 0 un suceso imposible, aunque realmente estos valores no se tendrán en cuenta, ya que un suceso imposible no supondría un verdadero riesgo y un suceso seguro sería un problema actual y no futuro.

A continuación se enumeran los riesgos con sus correspondientes descripciones, así como la descripción del impacto o efecto que podría tener sobre el proyecto y el plan de contingencia que se debería llevar a cabo para mitigar sus efectos.

Riesgo	Importancia	Probabilidad	RE
Estimación del tiempo incorrecta	5	9	45
Falta de conocimientos	4	7	28
Dificultades con el software de desarrollo	3	6	18
Toma de requisitos incorrecta o escasa	2	6	12
Cambios en las especificaciones	2	5	10
Problemas de ejecución en otros dispositivos	2	4	8
Dificultades de uso para los usuarios	2	4	8
Pérdida del proyecto	3	2	6
Ausencia del desarrollador	2	1	2

Tabla 2.2: Riesgos

Figura 2.35: Matriz de riesgos

Identificador	0001
Título	Estimación del tiempo incorrecta
Descripción del riesgo:	La planificación asignada al proyecto es demasiado optimista, y finalmente el tiempo necesario para llevar a cabo algunas tareas es mayor del estimado inicialmente.
Descripción del impacto:	Las tareas se demoran respecto a su planificación inicial.
Plan de contingencia:	Reorganización del plan en cuanto se detecte una demora respecto al plan inicial para evitar una desorganización mayor, reduciendo el tiempo de ejecución de otras tareas. Realizar un plan inicial pesimista respecto al tiempo, dado que se trata de un primer proyecto.

Identificador	0002
Título	Falta de conocimientos
Descripción del riesgo:	La alumna al tener poca experiencia práctica no tiene conocimientos suficientes para saber cómo actuar en determinadas situaciones.
Descripción del impacto:	Toma de malas decisiones durante el desarrollo del proyecto.
Plan de contingencia:	Solicitar ayuda a los miembros del equipo a la hora de tomar decisiones si no se está seguro de lo que hacer, antes de haberlo llevado a cabo.

Identificador	0003
Título	Dificultades con el *software* de desarrollo
Descripción del riesgo:	El *software* y las nuevas herramientas de desarrollo a utilizar son más complejas de lo que se esperaba o generan problemas al utilizarlas en el dispositivo de desarrollo.
Descripción del impacto:	Aumento en el tiempo necesario para realizar el proyecto y en el peor caso, imposibilidad para continuar con el desarrollo.
Plan de contingencia:	Solicitar ayuda a los miembros del equipo que tengan más experiencia con estas tecnologías.

Identificador	0004
Título	Toma de requisitos incorrecta o escasa
Descripción del riesgo:	Los requisitos recogidos con el cliente resultan insuficientes a la hora de desarrollar el producto.
Descripción del impacto:	No se pueden realizar ciertas tareas por falta de información o incertidumbre al iniciarlas.
Plan de contingencia:	Reunión con el cliente o *Product Owner* para obtener los requisitos que falten. Realizar una lista de las posibles dudas que puedan surgir con el desarrollo antes de hablar con el cliente.

Identificador	0005
Título	Cambios en las especificaciones
Descripción del riesgo:	Se ha desarrollado el producto conforme a unos requisitos que luego han sido modificados.
Descripción del impacto:	El producto no se adapta a lo que el cliente solicita por lo que es necesario modificarlo.
Plan de contingencia:	Buscar la forma de modificar el producto que más se adapte a lo ya realizado y también a las nuevas especificaciones. Procurar mantener una comunicación constante con el cliente para evitar que los cambios sean irreparables.

Identificador	0006
Título	Problemas de ejecución en otros dispositivos
Descripción del riesgo:	El producto funciona correctamente en el dispositivo de desarrollo, pero no en otros dispositivos donde se ha probado.
Descripción del impacto:	El producto es funcional pero solo en dispositivos con las mismas características que el de desarrollo, por lo que se reducen las posibilidades reales de uso.
Plan de contingencia:	Comprobar qué es lo que ocasiona este problema y adaptar el desarrollo para que el producto sea más flexible y se pueda adaptar a diferentes dispositivos.

Identificador	0007
Título	Dificultades de uso para los usuarios
Descripción del riesgo:	A la hora de probar el producto los usuarios no son capaces de utilizar la aplicación.
Descripción del impacto:	La utilidad del producto se ve mermada y no puede cumplir con su funcionalidad.
Plan de contingencia:	Realizar diferentes pruebas de usabilidad con los usuarios finales de la aplicación o con usuarios que cumplan con características similares a estos.

Identificador	0008
Título	Pérdida del proyecto
Descripción del riesgo:	El dispositivo de desarrollo desaparece o sufre algún accidente, perdiendo todo el proyecto que se tuviese solo en local.
Descripción del impacto:	Es necesario rehacer todo el proyecto que se haya perdido.
Plan de contingencia:	Mantener alguna copia del proyecto, a ser posible en la nube y actualizarla siempre que sea posible.

Identificador	0009
Título	Ausencia del desarrollador
Descripción del riesgo:	El alumno necesita ausentarse y no puede continuar con el desarrollo del proyecto durante alguna semana.
Descripción del impacto:	El proyecto permanece parado durante su ausencia.
Plan de contingencia:	Ser pesimistas en la planificación del proyecto para tener en cuenta posibles parones en su desarrollo y acelerar el proceso todo lo posible a la vuelta del desarrollador.

2.6. Presupuesto

En esta sección se pretende mostrar una estimación del coste que tendrá la realización del proyecto completo, desde los recursos materiales y *software*, hasta las horas de personal necesarias para llevar a cabo el desarrollo del sistema.

Los **recursos** utilizados para el desarrollo del proyecto incluyen el dispositivo utilizado para ello y el personal que ha participado en el mismo, así como las herramientas software. El dispositivo de desarrollo es un portátil HP Pavilion Gaming 15-EC0001NS valorado en unos 900€ en el momento de realización de este proyecto. Se ha de tener en cuenta que el uso del mismo se extiende más allá del propio proyecto, por lo que se estimará el coste de 12 meses (duración aproximada del proyecto) respecto a 5 años, ya que la vida útil media de un portátil suele estar entre los 3 y los 6 años.

En cuanto a los **miembros del equipo**, se destinarían los sueldos de cuatro personas a la realización de este proyecto, suponiendo que la alumna no estuviera realizando un Trabajo de Fin de Grado.

Si solo se cobran las horas estimadas, que serian unas 450 y que tres de los miembros del equipo solo dediquen unas 30 horas dado que su labor es ayudar en caso de que sea necesario y tutorizar el progreso del proyecto, las horas totales trabajadas serían unas 480.

Teniendo en cuenta la falta de experiencia de la alumna, se le asociará un sueldo diferente al del resto de desarrolladores. Suponiendo que el sueldo medio de un recién titulado está entre los 15.000 € y los 20.000 € y que se trabaje unas 1.830 horas al año, el sueldo medio por hora sería de: 17.500/1.830 = 9.5€/h

En cuanto al resto de desarrolladores, suponiendo que cobren en torno a los 23.000€ al año, el sueldo medio por hora sería de: 23.000/1.830 = 12.5€/h.

Respecto a las **herramientas** *software* utilizadas: Visual Studio Code, PhpStorm, Astah, Figma y Slack, se utiliza una versión gratuita salvo en el caso de PhpStorm, en el que se ha utilizado la licencia de estudiante, y cuyo valor real sería de 200€. Por su parte, de GitHub se utiliza en su versión de empresa con un valor de 18€ al mes aprox. Aun así, en el cálculo de los costes se tendrán en cuenta las versiones de pago, salvo en el caso de Slack, ya que la empresa utiliza la versión gratuita.

Las cuentas realizadas a partir de esta información se muestran en la **Figura 2.36** e indican que el presupuesto total sería de 5691 € considerando el uso de las versiones de pago de las herramientas y que el resto de hipótesis sean correctas.

FUENTE	VALOR (€)	CUENTAS	GASTO (€)
Portátil HP	900	Se usará 1 año de los 5 de vida útil media	180
Desarrollador Junior	9,5	Sueldo medio por hora multiplicado por las 450 horas del proyecto	4275
Resto desarrolladores	12,5	Sueldo medio por hora multiplicado por 30 horas añadidas al proyecto	375
Visual Studio Code	35	Valor de la licencia de la versión de pago	35
Astah	290	Valor de la licencia de la versión de pago	290
PhpStorm	200	Coste real de la licencia de estudiante	200
Figma	10	Coste mensual de la versión de pago multiplicado por 12 meses	120
GitHub	18	Coste de la versión de empresa al mes multiplicado por 12 meses	216
TOTAL			5691

Figura 2.36: Presupuesto

Capítulo 3

Análisis y diseño

Este capítulo se divide en dos secciones, una de **análisis** y otra de **diseño**, en las que se va a presentar lo siguiente:

- Los **requisitos** obtenidos a partir de las *user stories* y los **casos de uso**.
- Los **modelos** y **diagramas** de análisis y diseño desarrollados.
- El diseño de la **interfaz**.

3.1. Análisis

3.1.1. Requisitos

Los requisitos son las condiciones que debe cumplir o poseer el sistema o uno de sus componentes para satisfacer un contrato, norma o especificación [10]. A continuación se describen los requisitos establecidos a partir de las *user stories* explicadas en el **Capítulo 2**:

Requisitos funcionales

Se conocen como requisitos funcionales aquellos que describen los servicios que el sistema debe proporcionar: cómo debe reaccionar a una entrada particular o ante situaciones concretas. Es decir, definen el funcionamiento del sistema.

Los requisitos funcionales relacionados con los **registros** de las horas trabajadas son:

- **RF1.** El sistema debe permitir al usuario registrar sus horas de entrada y salida.
- **RF2.1. - Visualización.** El sistema debe permitir al usuario acceder a sus registros.

- **RF2.2. - Visualización.** El sistema debe permitir al usuario con permisos de administrador acceder a los registros del resto de usuarios.
- **RF3.1. - Creación.** El sistema debe permitir al usuario registrar las horas en el momento en que inicie o termine la actividad correspondiente. Es decir, cuando la hora de registro se corresponda con la hora registrada.
- **RF4.1. - Modificación.** El sistema debe permitir al usuario modificar sus registros.

Los requisitos funcionales relacionados con los **reportes** son:

- **RF5.** El sistema debe permitir al usuario con permisos de administrador obtener un reporte sobre los datos registrados en la aplicación.
- **RF5.1.** El sistema debe permitir a todos los usuarios acceder a un resumen visual de sus horas trabajadas y eventos próximos.

Los requisitos funcionales relacionados con las **vacaciones o periodos de inactividad laboral** son:

- **RF6.** El sistema debe permitir al usuario solicitar períodos de vacaciones o inactividad laboral.
- **RF6.1.** El sistema debe permitir a los usuarios ver aquellos eventos que tengan permisos de visibilidad activados.
- **RF6.2.** El sistema debe permitir al usuario que haya solicitado el período de inactividad, modificar su visibilidad, de forma que se oculte al resto de usuarios, salvo a aquel(los) con permisos de administrador.

Otros requisitos funcionales son:

- **RF7.** El sistema debe permitir a los usuarios acceder a las funcionalidades identificándose mediante un nombre de usuario/dirección e-mail y una contraseña.
- **RF8.** El sistema debe permitir al usuario con permisos de administrador modificar, añadir y/o eliminar a cualquier usuario que no tenga permisos de administrador.

Requisitos de información

Los requisitos de información son una forma especial de requisitos funcionales que indican el tipo de información que guarda el sistema.
Los requisitos de información relacionados con los **roles de usuario** son:

- **RI1.** Un usuario podrá clasificarse según diferentes categorías: por el rol que presenten dentro de la empresa, por tratarse o no de un usuario con permisos de administrador.

- **RI1.1.** Se guardará información sobre el rol o función que tenga cada usuario dentro de la empresa, de forma que el sistema pueda saber si dos usuarios con el mismo rol han solicitado un período de inactividad en las mismas fechas.

Los requisitos de información relacionados con los **registros** son:

- **RI2.** Se diferenciará entre tiempo de actividad del empleado, tiempo de descanso, y tiempo de inactividad.

Los requisitos de información relacionados con los **reportes** son:

- **RI3.** La información que se incluirá en el reporte mostrará para cada entrada: el identificador del usuario correspondiente, la fecha, las horas de entrada y salida que delimiten las horas activas del usuario, las horas de inicio y final de los descansos realizados y las horas totales trabajadas esa fecha.

Requisitos no funcionales

Se conocen como requisitos no funcionales a aquellos que afectan a los servicios o funciones del sistema, como restricciones de tiempo, sobre el proceso de desarrollo, estándares...Definen propiedades emergentes del sistema, necesidades técnicas o legales y pueden ser incluso más críticos que los requisitos funcionales.
Los requisitos no funcionales relacionados con los **reportes** son:

- **RNF1.** El formato del reporte podrá ser un archivo .pdf o .csv.

Requisitos de usabilidad

Los requisitos de usabilidad son requisitos no funcionales orientados a mejorar la interacción del usuario con el sistema. Se han establecido los siguientes:

- **RU1.** El sistema debe permitir a los usuarios crear sus registros de trabajo y descanso de forma eficiente, en un número de pasos inferior a 4.
- **RU2.** El sistema debe permitir a los usuarios modificar y/o eliminar sus registros de forma sencilla y fácil de aprender. Debe poder realizarse en menos de un minuto en el primer intento.
- **RU3.** El sistema debe mostrar a los usuarios un resumen gráfico de sus registros que resulte fácil de comprender. Los usuarios deben poder extraer la información correcta en menos de un minuto.

3.1.2. Casos de uso

En esta sección se presentan los actores y casos de uso que conforman el funcionamiento principal del sistema a desarrollar. En primer lugar, un **caso de uso** [25] [26] es un conjunto de escenarios exitosos o no, relacionados entre sí por el resultado que el actor espera obtener.

Para entender esta definición hay que conocer lo que son un escenario y un actor [26]. Un **escenario** es una secuencia de acciones del sistema que conducen a un resultado. Un **actor** es cualquier entidad que se comunica con el sistema, pudiendo ser un actor principal si hace uso de los servicios del sistema, un actor de apoyo, en caso de ser el que proporciona servicios al sistema o un actor pasivo si simplemente está interesado en el comportamiento del caso de uso.

En este sistema existen dos actores principales:

- **Empleado:** Cualquier usuario que pueda acceder a la aplicación, identificarse y utilizar sus servicios para mantener un registro de su jornada laboral.
- **Superusuario o admin:** Aquel usuario que, además de poder accede a la aplicación e identificarse, estará encargado de la administración de la misma. Se trata de una especialización del anterior.

A continuación se describen los casos de uso que establecen el flujo de interacción de ambos actores con el sistema. Para tener una visión más global de los casos de uso que se van a explicar, se presenta el **diagrama de casos de uso** en la **Figura 3.1**:

Diagrama de casos de uso

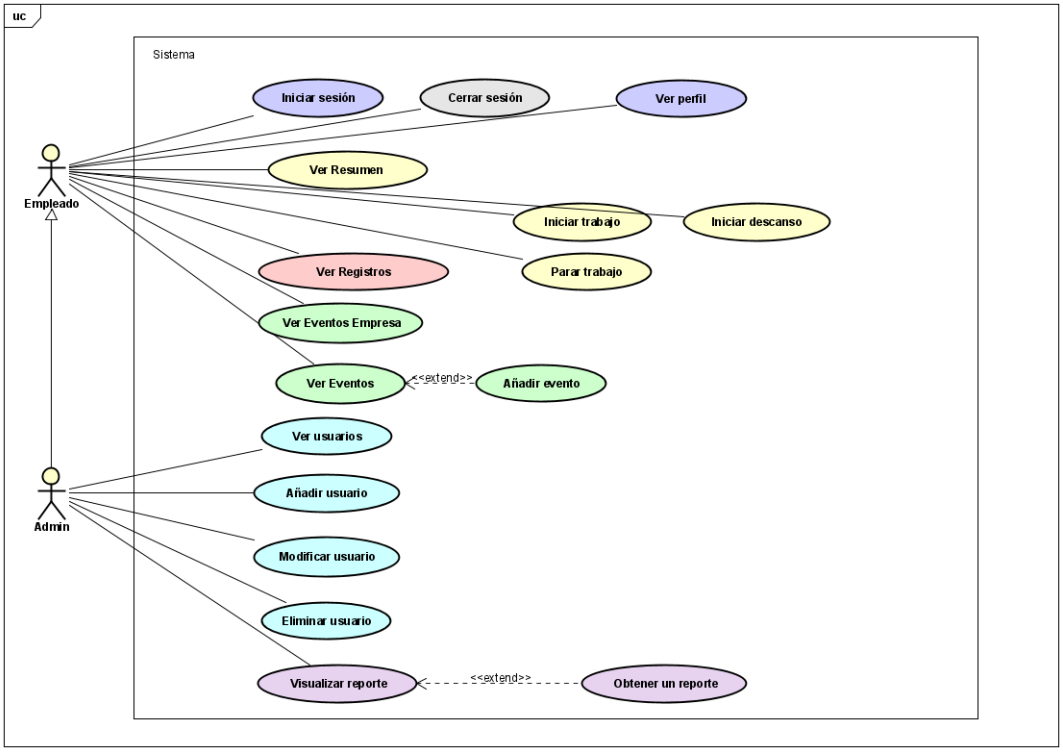


Figura 3.1: Diagrama de casos de uso

Como puede verse en este diagrama, el usuario **Admin** puede realizar las mismas acciones que el usuario **Empleado** y además, puede gestionar los usuarios del sistema y obtener reportes elaborados a partir de los datos o registros de los mismos.

Además, en el caso de los eventos, se permitirá al usuario que, cuando visualice los propios, pueda añadir eventos nuevos. Por otro lado, la visualización de los reportes permitirá a su vez generar un fichero .pdf o .csv con la totalidad de los datos mostrados.

CASO DE USO: Iniciar sesión

Actor: Empleado.

Precondición: El actor no ha iniciado sesión y no se encuentra identificado en el sistema.

Secuencia normal:

1. El actor introduce el e-mail y la contraseña.
2. El sistema comprueba que dicho par (e-mail, contraseña) se encuentran en la base de datos del sistema y muestra la vista del *Dashboard*.

Alternativas y excepciones:

2. -a. El e-mail introducido no está registrado en el sistema. Se muestra un mensaje de error y el caso de uso queda sin efecto.
2. -b. La contraseña no se corresponde con la correspondiente al e-mail introducido. Se muestra un mensaje de error y el caso de uso queda sin efecto.

Postcondición: Escenario de éxito: El usuario está identificado en el sistema, se muestra la vista del *Dashboard* y puede acceder al resto de secciones.

CASO DE USO: Cerrar sesión

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema cerrar su sesión.
2. El sistema limpia la información de sesión de usuario y lo redirige a la página de inicio de sesión.

Postcondición: Escenario de éxito: El usuario no está identificado en el sistema y se encuentra en la vista del *Login*.

CASO DE USO: Ver perfil

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema visualizar los datos de su usuario.
2. El sistema muestra los datos del usuario loggeado en la vista correspondiente.

Postcondición: Escenario de éxito: El usuario puede ver sus datos.

CASO DE USO: Ver Resumen

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema un resumen de sus registros.
2. El sistema muestra la vista correspondiente y carga la información de los registros del usuario loggeado.

Postcondición: Escenario de éxito: El usuario puede ver un resumen de sus registros.

CASO DE USO: Iniciar trabajo

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema iniciar un registro de trabajo.

2. El sistema comprueba que la acción solicitada es válida.

3. El sistema crea un registro de *trabajo*.

Alternativas y excepciones:

2. -a. Se ha solicitado crear un registro de *trabajo*, pero ya se está ejecutando un registro de trabajo. Se crea un registro de *descanso*.

Postcondición: Escenario de éxito: El sistema ha creado un registro de *trabajo* y la vista se ha modificado de forma que el usuario sepa que esto ha ocurrido.

CASO DE USO: Iniciar descanso

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema iniciar un registro de descanso.

2. El sistema comprueba que la acción solicitada es válida.

3. El sistema crea un registro de *descanso*.

Alternativas y excepciones:

2. -a. Se ha solicitado crear un registro de *descanso*, pero ya se está ejecutando un registro de descanso. Se crea un registro de *trabajo*.

Postcondición: Escenario de éxito: El sistema ha creado un registro de *descanso* y la vista se ha modificado de forma que el usuario sepa que esto ha ocurrido.

CASO DE USO: Parar trabajo

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema parar un registro de trabajo.
2. El sistema comprueba que la acción solicitada es válida.
3. El sistema finaliza el registro de *trabajo*.

Alternativas y excepciones:

2. -a. Se ha solicitado parar un registro de *trabajo*, pero el que se está ejecutando es de descanso. No se permite la detención.

Postcondición: Escenario de éxito: El sistema ha parado el registro de *trabajo* y la vista se ha modificado de forma que el usuario sepa que esto ha ocurrido.

CASO DE USO: Ver Registros

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema visualizar sus registros.
2. El sistema muestra la vista correspondiente y carga la información de los registros del usuario loggeado.

Postcondición: Escenario de éxito: El usuario puede ver sus registros en la vista correspondiente.

CASO DE USO: Ver Eventos

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema visualizar sus eventos.
2. El sistema muestra la vista correspondiente y carga la información de los eventos del usuario loggeado.

Postcondición: Escenario de éxito: El usuario puede ver sus eventos.

CASO DE USO: Añadir Evento

Actor: Empleado.

Precondición: El actor ha iniciado sesión, se encuentra identificado en el sistema y ha visualizado sus eventos.

Secuencia normal:

1. El actor solicita al sistema añadir un nuevo evento.
2. El sistema muestra el formulario para añadir un nuevo evento.
3. El actor rellena el formulario con la información correspondiente al nuevo evento.
4. El sistema comprueba que los datos son correctos.
5. El sistema añade el nuevo evento.

Alterantivas y excepciones:

4. -a. Los datos del evento interfieren con los de otro usuario y el sistema muestra un mensaje de error.

Postcondición: Escenario de éxito: Se añade el nuevo evento a los eventos del usuario loggeado.

CASO DE USO: Ver Eventos Empresa

Actor: Empleado.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema ver los eventos del resto de usuarios.
2. El sistema muestra la vista correspondiente a y carga la información de los eventos de los usuarios correspondientes.

Postcondición: Escenario de éxito: El usuario puede ver los eventos de los usuarios que así lo permitan.

CASO DE USO: Ver Usuarios

Actor: Admin.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema ver los usuarios registrados en el mismo.
2. El sistema muestra la la información de los usuarios.

Postcondición: Escenario de éxito: El usuario ve la información correspondiente a todos los usuarios del sistema.

CASO DE USO: Añadir usuario

Actor: Admin.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema añadir un usuario.
2. El sistema muestra el formulario para añadir los datos del nuevo usuario.
3. El actor introduce los datos correspondientes al nuevo usuario.
4. El actor indica al sistema que ha terminado de rellenar la información.
5. El sistema comprueba que los datos son correctos.
6. El sistema crea el nuevo usuario.

Alternativas y excepciones:

5. -a. Los datos introducidos ya corresponden a otro usuario y el sistema muestra un mensaje de error.
5. -b. Los datos introducidos son insuficientes o incorrectos y el sistema muestra un mensaje de error.

Postcondición: Escenario de éxito: El sistema ha creado el nuevo registro de usuario.

CASO DE USO: Modificar usuario

Actor: Admin.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema modificar un usuario.
2. El sistema muestra el formulario para modificar los datos del usuario correspondiente.
3. El actor modifica los datos correspondientes al usuario.
4. El actor indica al sistema que ha terminado de rellenar la información.
5. El sistema comprueba que los datos son correctos.
6. El sistema actualiza los datos del usuario.

Alternativas y excepciones:

5. -a. Los datos introducidos son incorrectos y el sistema muestra un mensaje de error.

Postcondición: Escenario de éxito: El sistema ha actualizado el registro de usuario.

CASO DE USO: Eliminar usuario

Actor: Admin.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita al sistema eliminar un usuario.
2. El sistema elimina los datos del usuario.

Postcondición: Escenario de éxito: El sistema ha eliminado el registro de usuario.

CASO DE USO: Visualizar reporte

Actor: Admin.

Precondición: El actor ha iniciado sesión y se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita visualizar los reportes de los registros.
2. El sistema muestra la información de los usuarios del sistema y sus registros.

Postcondición: Escenario de éxito: El usuario puede ver los reportes generados a partir de los registros y datos de los usuarios.

CASO DE USO: Obtener un reporte

Actor: Admin.

Precondición: El actor ha iniciado sesión, se encuentra identificado en el sistema y está en la vista de visualización de reportes.

Secuencia normal:

1. El actor selecciona las opciones de reporte que desee.
2. El actor solicita al sistema obtener el reporte indicado.
3. El sistema devuelve el archivo correspondiente.

Postcondición: Escenario de éxito: El actor obtiene el reporte con las opciones seleccionadas.

3.1.3. Modelo de dominio

El **modelo de dominio** [27] [28] pretende ser una representación de conceptos y relaciones entre los mismos, un marco de referencia conceptual que permita ver de forma abstracta el contexto o dominio del problema que se va a tratar y que facilite la traducción a un modelo específico del sistema gestor de base de datos.

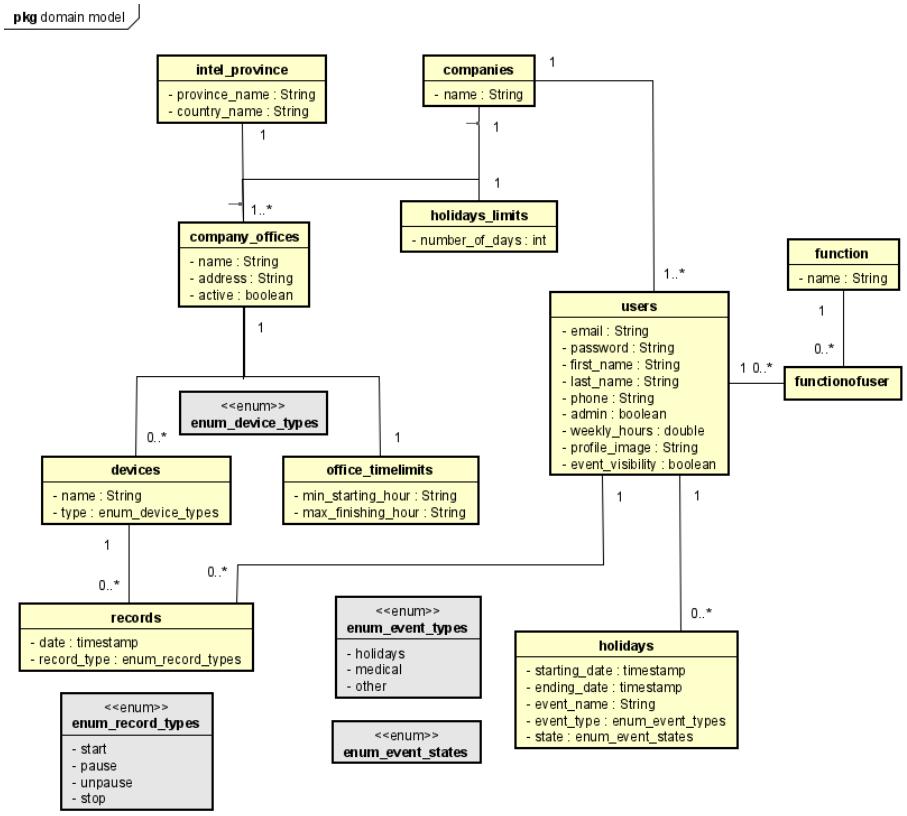


Figura 3.2: Modelo de dominio

Como puede verse en la **Figura 3.2**:

- Un **usuario** puede tener cero o más **eventos** o **vacaciones**, así como cero o más **funciones**. Del mismo modo, puede crear cero o más **registros**.
- Los **registros** conceptualmente estarán asociados a un dispositivo, que a su vez pertenecerá a una **oficina**. Esta **oficina** estará situada en una **provincia** y pertenecerá a una **compañía**.
- Además, las **oficinas** tendrán unos límites de horario para iniciar y terminar la jornada y las **compañías** tendrán un límite de días que se puedan solicitar como **vacaciones**.

- Tanto los dispositivos, como las oficinas, provincias, compañías y sus respectivos horarios se tendrán en cuenta para la composición de la base de datos, pero su gestión se mantendrá al margen de la funcionalidad de la aplicación.

3.1.4. Diagramas de secuencia

Un **diagrama de secuencia** [29] muestra los eventos que generan los actores y el sistema, y el orden en que se generan para un escenario específico de un caso de uso. Dada la similitud entre los casos de uso para acceder a una sección y entre los casos de uso para crear registros de diferentes tipos, se detallarán solo algunos de los casos de uso explicados en las secciones anteriores.

Así, se muestra en la **Figura 3.3** el diagrama de secuencia para el caso de uso *Iniciar Sesión*:

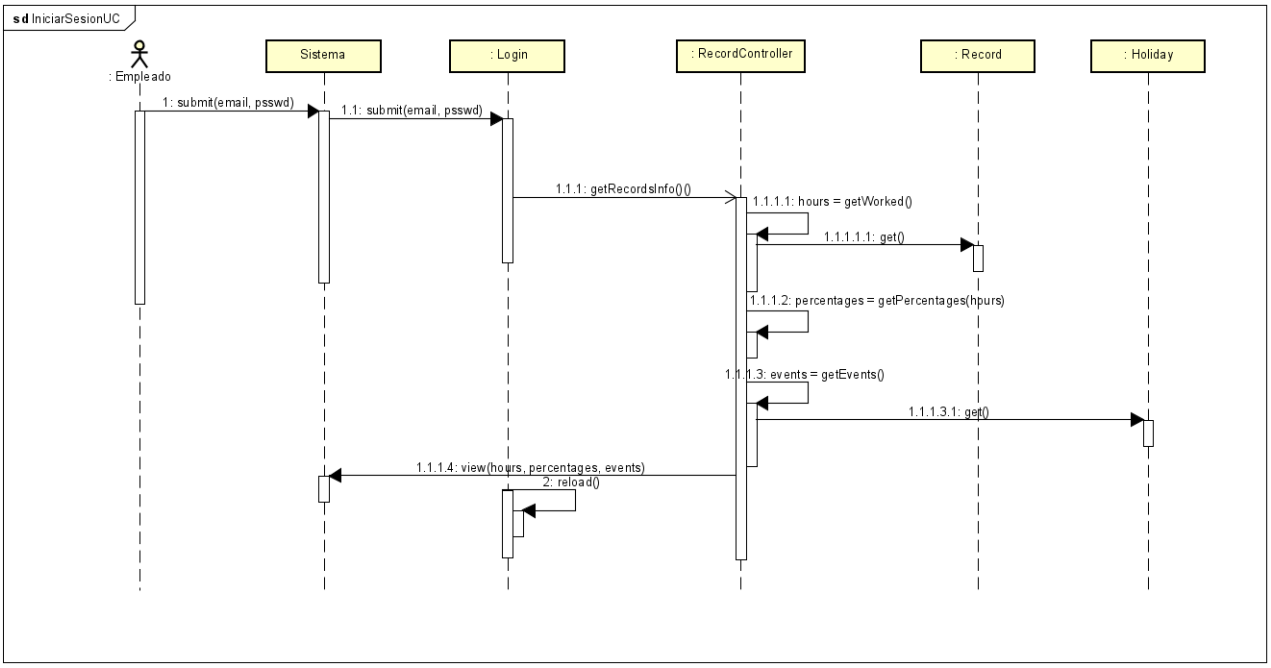


Figura 3.3: Diagrama de secuencia para *Iniciar Sesion*

Para ejemplificar la creación de registros se muestra el diagrama de secuencia correspondiente al inicio, pausa y reinicio de un registro de trabajo, dejando por desarrollar el diagrama de secuencia para detener un registro de trabajo, que sería similar a lo mostrado en la **Figura 3.4**.

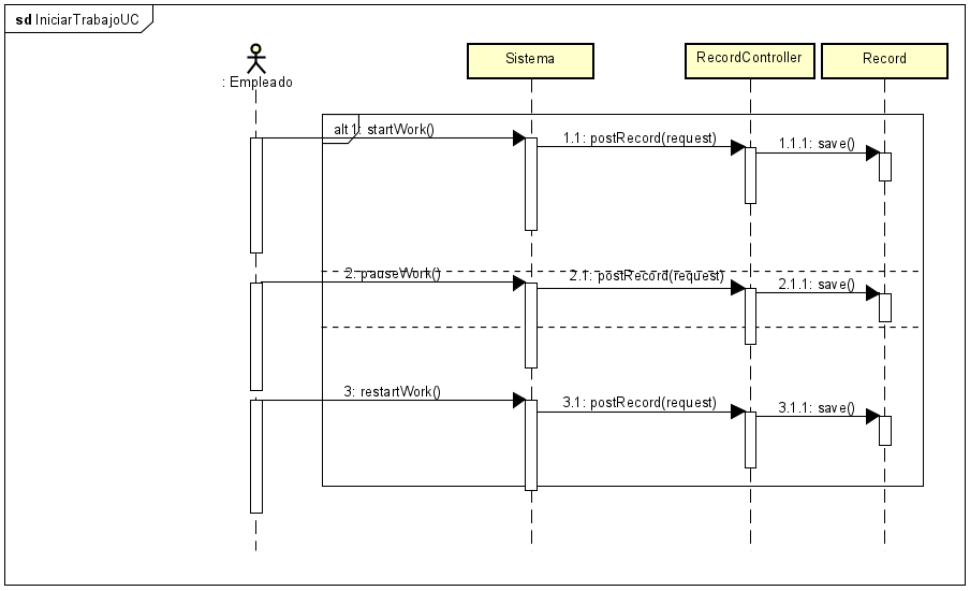


Figura 3.4: Diagrama de secuencia para *Iniciar Trabajo*

En la **Figura 3.5** se muestra el diagrama de secuencia correspondiente a añadir un usuario, que muestra un comportamiento similar al de los casos de uso en que se modifica un usuario o se añade un evento.

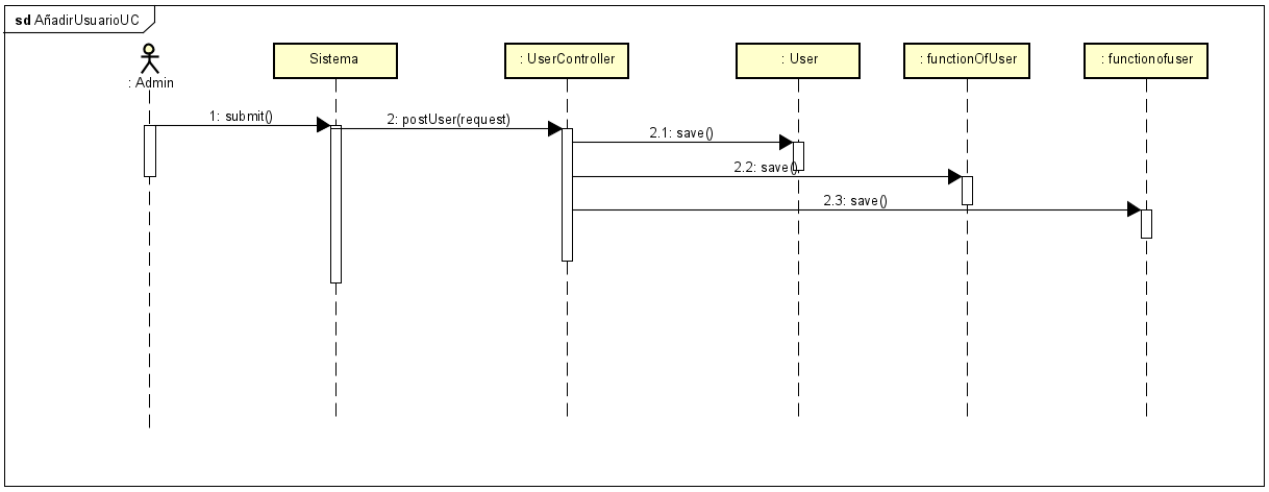


Figura 3.5: Diagrama de secuencia para *Añadir Usuario*

Por último, en la **Figura 3.6** se muestra el diagrama de secuencia para visualizar o acceder a información, en concreto la correspondiente a la información del usuario que ha iniciado sesión. Para visualizar otra información como los registros o eventos, el proceso sería similar, modificando los datos según lo solicitado por el usuario. Esta información se solicitaría a los diferentes controladores correspondientes a los conceptos representados en el modelo de dominio.

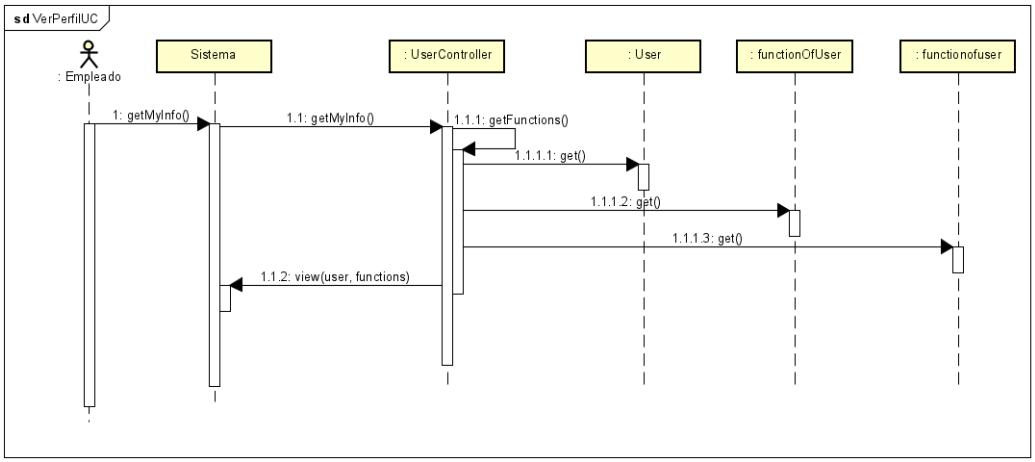


Figura 3.6: Diagrama de secuencia para *Ver Perfil*

3.2. Diseño

3.2.1. Arquitectura

El ***Decomposition Style*** [30] se utiliza para descomponer un sistema en unidades más pequeñas de implementación, conocidas como módulos y submódulos. Estas permiten describir la organización del código.

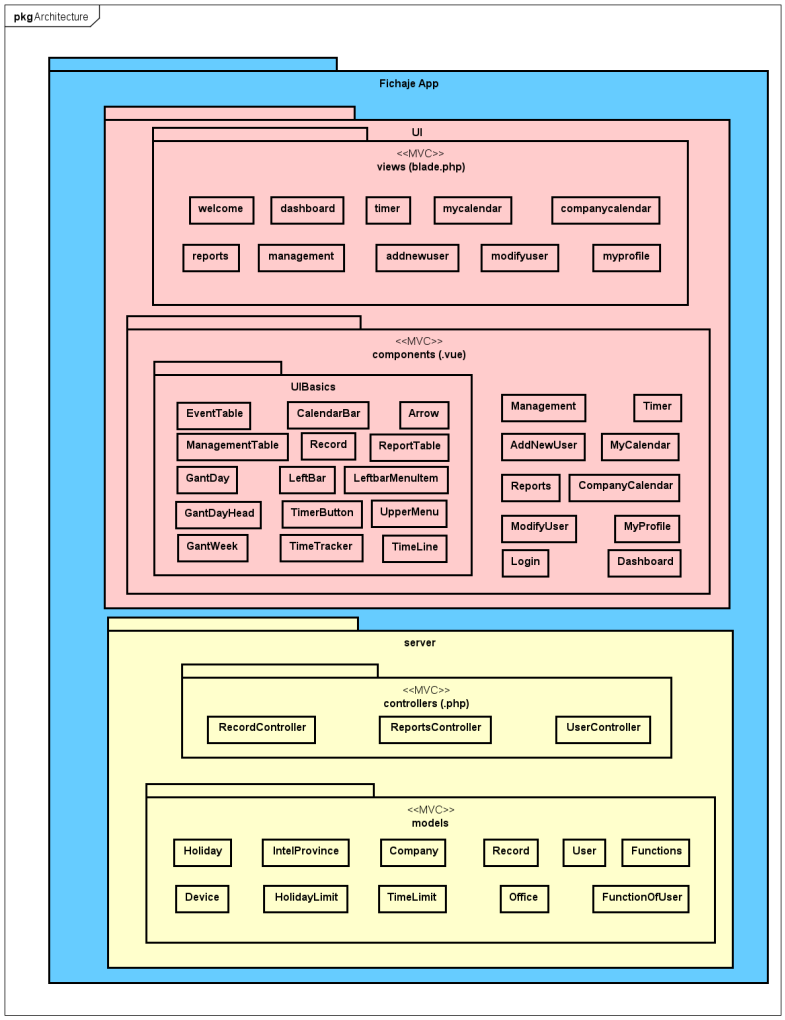


Figura 3.7: Arquitectura *Decomposition Style*

El motivo para optar por esta representación se basa en que el *framework* utilizado, *Laravel*, ya proporciona una estructura básica de desarrollo separando las vistas de los controladores y modelos. Esto puede verse reflejado en la **Figura 3.7**, donde se separan los módulos de la interfaz de los

correspondientes a los controladores y modelos. Dentro del módulo correspondiente a la interfaz, se puede establecer otra división, en las vistas propias de *Laravel*, con la extensión *blade.php* y las componentes que conformarán estas vistas, desarrolladas con el *framework Vue*.

Dentro de estas vistas encontramos una correspondencia entre las vistas de *Vue* y las de *Laravel*, que a su vez se asocian a algunos de los casos de uso; como es el caso de las vistas de *welcome* y *Login* con el caso de uso *Iniciar sesión*. las vistas homónimas *dashboard* para el caso de uso *Ver Resumen*, las vistas *timer* para el caso de uso *Ver Registros* y las vistas *mycalendar* y *companycalendar* para los casos de uso *Ver Eventos* y *Ver Eventos Empresa* o *reports* para *Visualizar reporte*. Lo mismo ocurre con *addnewuser* y *modifyuser* para los casos de uso *Añadir usuario* o *Modificar usuario*.

Por su parte, los modelos y controladores se agrupan en el módulo del servidor y dentro de este se dividen en controladores y modelos, siendo estos últimos la representación de la información contenida en cada una de las tablas de la base de datos y a los que accederán los controladores para extraer o modificar información de las mismas. Estos controladores se han agrupado en tres principales que serán suficiente para acceder a toda la información mostrada en la aplicación: *RecordController*. *ReportsController*, que accede a información de varios modelos al mismo tiempo y *UserController*.

3.2.2. Diseño de las vistas e interacciones (UI|UX)

En esta sección se explican las decisiones tomadas respecto a ciertos aspectos de la interfaz, así como el funcionamiento de las interacciones entre unas y otras vistas.

En primer lugar y tomando como inspiración el sitio web de la empresa cliente [31] y la temática de las actividades realizadas en la misma, se opta por buscar una paleta de colores relacionada con la tierra, extraída de la referencia [32] y concretamente la que se observa en la **Figura 3.8**.

Tras elegir los colores protagonistas de la aplicación, se decide basar el conocido como imagotipo (conjunto de logotipo e isotipo o icono propios de la marca) de la misma, en el que ya presenta la empresa cliente y que se muestra en la **Figura 3.9**:

Así, el imagotipo de la aplicación presenta el aspecto de la **Figura 3.10**, donde se opta por utilizar el mismo isotipo, es decir, el icono representativo de la marca, pero modificando su diseño de forma que el círculo central muestre un reloj de aguja y los círculos más pequeños recuerden a un botón de pausa y un botón de inicio, asociándolo así a la empresa cliente y a la funcionalidad principal de la aplicación web, que será registrar la jornada laboral o las horas de trabajo de los usuarios.

En cuanto al logotipo, se utiliza la misma tipografía que la del logotipo de Biome Makers, pero cambiando el contenido. Concretamente, el nombre de la aplicación se basará en un juego de palabras obtenido a partir del nombre de la empresa cliente y la expresión “llegar a tiempo” en inglés o “Be on time”. De este modo, el nombre de la aplicación será: “Bion time”.

A continuación, en la **Figura 3.11**, se muestra el diseño de la vista de *Login*, cuyo formato se basa en el *login* de otras herramientas utilizadas por los usuarios de la empresa. En concreto, se mantiene la proporción entre la imagen de fondo y la sección de la pantalla donde se solicita el usuario y la contraseña. Este parecido con otras aplicaciones ya utilizadas por los que van a ser

Figura 3.8: Paleta de colores elegida

Figura 3.9: Logo de Biome Makers

los usuarios finales, tiene como objetivo que la interacción con la misma les resulte familiar y así, facilitar el uso de la aplicación reduciendo la complejidad al usarla por primera vez.

Se busca mantener la coherencia con el resto de la aplicación web, mostrando el isotipo sobre la imagen de fondo y utilizando la paleta de colores elegida en el botón y los campos del formulario, así como en el mensaje para recuperar la contraseña.

Figura 3.10: Logo de Bion Time

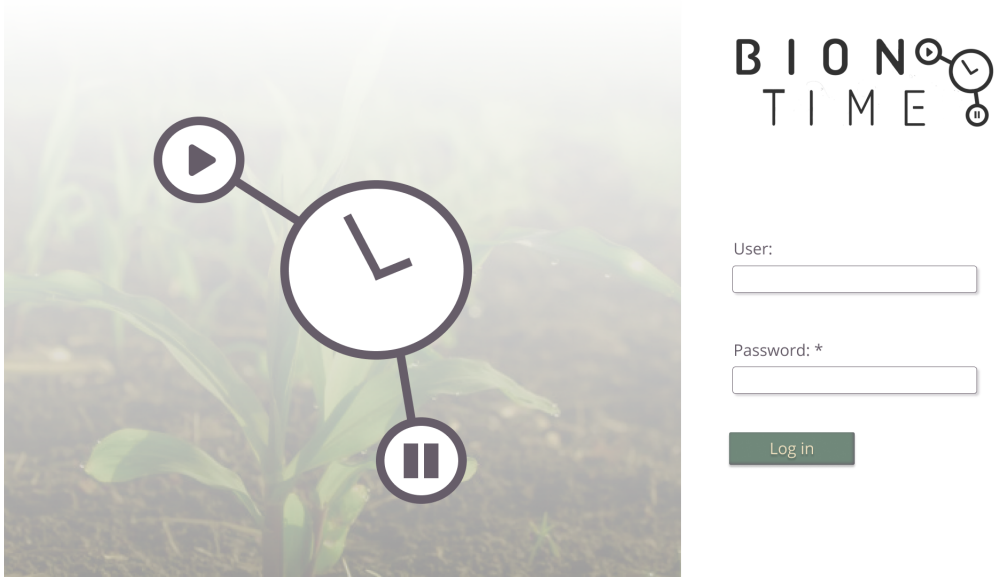


Figura 3.11: Vista de Login

En la vista de ***Summary***, así como en el resto de vistas, a excepción de la vista de *Login*, se sigue una estructura similar que, aparte de simplificar el funcionamiento de la aplicación al usuario, permite hacer visible el estado de la misma al establecer un borde del mismo color que las etiquetas correspondientes a cada sección alrededor de la relativa a la vista en la que se encuentra el usuario, como puede observarse, por ejemplo, con la etiqueta *Summary* en el menú izquierdo de las **Figuras 3.12, 3.13, 3.14 y 3.15**.

Además de esto, se mantiene una consistencia externa con otras aplicaciones, ya que muchas otras presentan menús laterales similares. Entre estas otras aplicaciones, una de las utilizadas como guía para el diseño de las vistas, *Toggl*. Este menú, así como el resto de contenido, tiene en cuenta que los usuarios web se fijan principalmente en la zona izquierda, superior e intermedia, formando una F. Por ello, el menú que lleva a las diferentes vistas se sitúa preferentemente a la izquierda.

Tanto en las vistas correspondientes a la sección *Summary* como las que se muestran en la sección *Timer* se muestra una barra o menú superior donde se encuentra el botón correspondiente

a la funcionalidad principal, que es crear registros de la jornada laboral. Este botón se divide en tres, donde el de mayor tamaño corresponde a la acción más realizada de las que permite llevar a cabo este, es decir, los botones de *play* y *pause*. La detención de los registros de trabajo se realiza con el botón superior y el cierre del sistema se encuentra en el botón inferior, ambos de menor tamaño. Siguiendo la ley de Fitts, se sitúan cerca funciones similares y se da el mayor tamaño a la acción principal.

El contenido de la vista de *Summary* permite ver dos gráficos circulares que muestran la proporción, así como el total de horas trabajadas durante el día y la semana actual respecto de las planeadas (comúnmente, 8 y 40 horas respectivamente).
Además de esto, se muestra una tabla con los eventos próximos correspondientes al usuario loggeado.

Figura 3.12: Vista de Summary

Dentro de las interacciones con la barra superior se encuentra la posibilidad de consultar qué horas se ha trabajado, bien durante el día, la semana o el mes actual. Para ello se despliega un menú situado en la parte superior izquierda, donde se pueden seleccionar estas tres opciones, como se muestra en la **Figura 3.13**. El resultado de seleccionar la semana se muestra en la **Figura 3.14** y el de seleccionar el mes en la **Figura 3.15**. Además de dichas interacciones, esta barra podrá desplegarse o plegarse para disminuir el espacio que ocupa en la pantalla.

En las vistas correspondientes al ***Timer***, es decir, aquellas en las que se muestran los registros de la jornada laboral, el contenido, además de la barra superior, que es igual que la mostrada en la vista de *Summary* y que muestra un contador del registro que va a ser añadido a la base de datos y los botones necesarios para esto; presenta la lista de registros total o bien los registros correspondientes a un día o una semana. Esto último depende de la vista seleccionada, que se

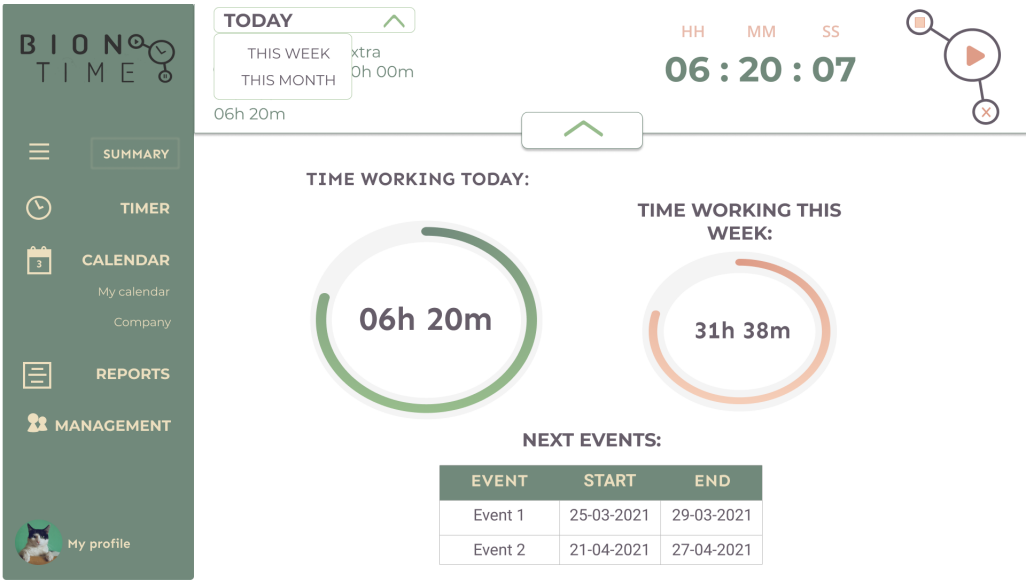


Figura 3.13: Vista de Summary

puede elegir de una lista desplegable semejante a la de la barra superior, en la parte superior derecha, bajo la barra superior.

Para facilitar el aprendizaje se siguen principios de diseño como el mapeado natural, en concreto mediante un código de colores asociados a los tiempos de trabajo, descanso y tiempo fuera del trabajo.

Esto se aplica a los registros mostrados en las vistas ***Timer List***, en la **Figura 3.16**, ***Timer Daily***, en la **Figura 3.17** y ***Timer Weekly***, en la **Figura 3.18**.

Además, en la vista *Timer List*, se aplica a las barras donde se muestra la proporción correspondiente a cada tiempo, bajo la barra superior. Posteriormente, tras analizar más en profundidad el funcionamiento de la aplicación, se añadirá, durante la implementación, al contador de la barra superior un reborde del color asociado al tipo de registro cuyo valor está siendo registrado, para facilitar al usuario la comprensión y visibilidad del estado del sistema. De este modo, las acciones del usuario reciben un *feedback* más inmediato, ya que cuando haya demasiados registros en la página le será más difícil comprobar que, efectivamente, hay uno nuevo y que este es el correspondiente a la fecha y hora deseadas.

En la **Figura 3.19** se muestra la vista correspondiente al calendario individual o ***My Calendar***, donde se presentan los eventos en una barra superior similar a la que se encuentra en las vistas de *Summary* y *Timer*. En ésta, además del calendario donde se podrán ver por un código de colores los diferentes eventos del usuario correspondiente, se permitirá crear un nuevo evento, como se muestra en la **Figura 3.21**.

En la **Figura 3.20**, se muestra el resultado de pasar el cursor por el botón correspondiente a la adición de un evento nuevo. Se muestra una *tooltip* que proporciona una pista de lo que se

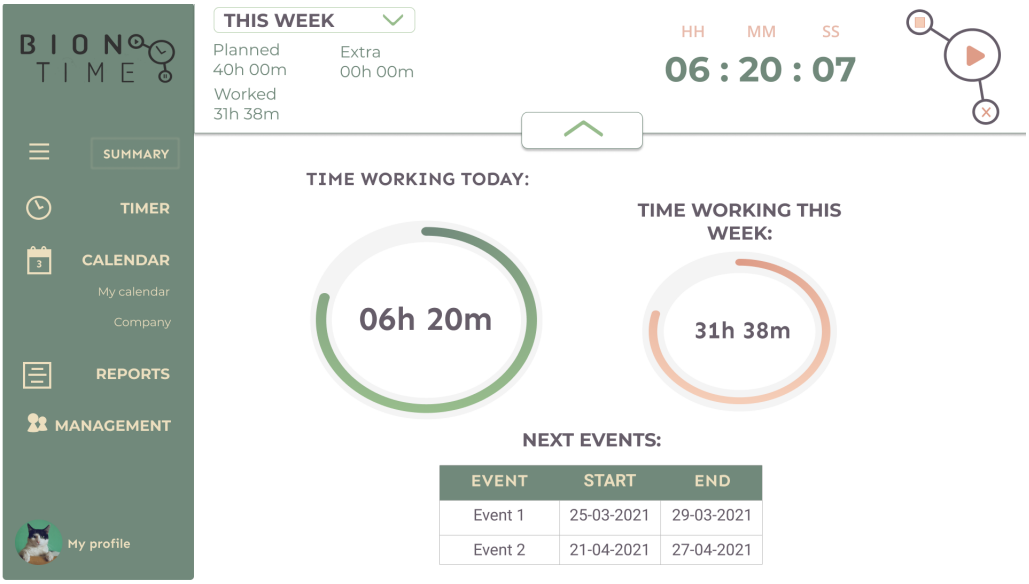


Figura 3.14: Vista de Summary

desplegará al pulsar ese botón. Este tipo de "pistas" se utilizarán también en otros elementos para facilitar la comprensión del sistema, junto con cambios en el cursor al posicionarlo sobre elementos con los que se pueda interactuar.

El calendario global o de la compañía se muestra en la vista de ***Company***, en la **Figura 3.22**, donde se muestra una tabla con las vacaciones o bajas de los diferentes usuarios que permitan su visualización. Además, se permite filtrar por aquellos usuarios que tengan una función parecida utilizando el botón central.

En la sección de ***Reports***, vista que se muestra en la **Figura 3.23**, se presenta una tabla con los registros de usuarios y varios campos relativos a estos, que podrán filtrarse o extraerse a diferentes formatos.

Un filtro similar al de la vista de *Reports* se utiliza en la vista ***Management***, en la **Figura 3.24**, donde se muestran los usuarios del sistema y donde se puede añadir uno nuevo pasando a la vista de ***Add New User*** en la **Figura 3.25**, en la que se presentará un formulario con diferentes valores.

Estas dos últimas vistas, las de *Management* y *Reports*, serán solo visibles para el administrador.

Por último, la vista de ***My Profile***, en la **Figura 3.26**, se muestran los detalles del usuario actual y su imagen de usuario.

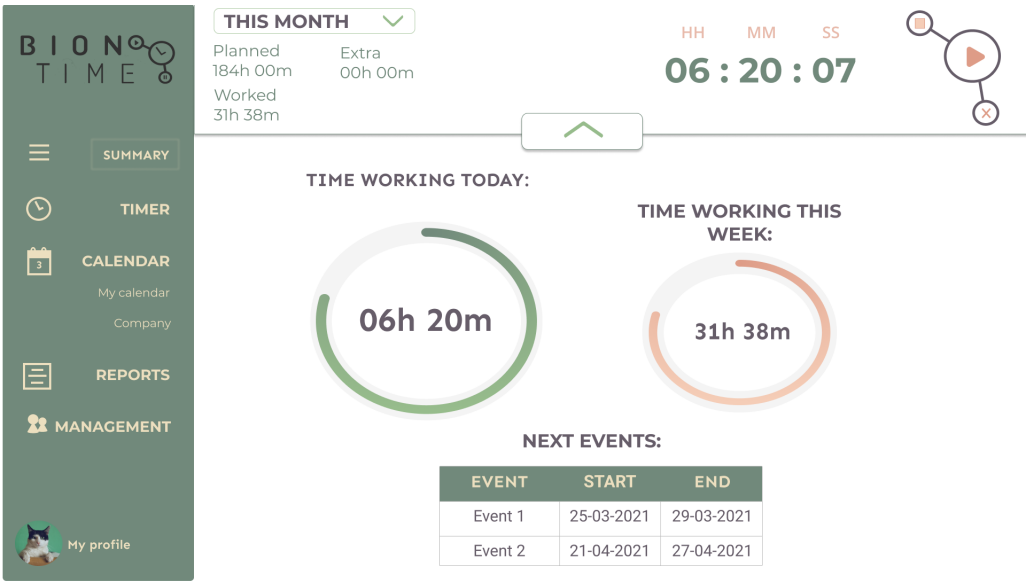


Figura 3.15: Vista de Summary

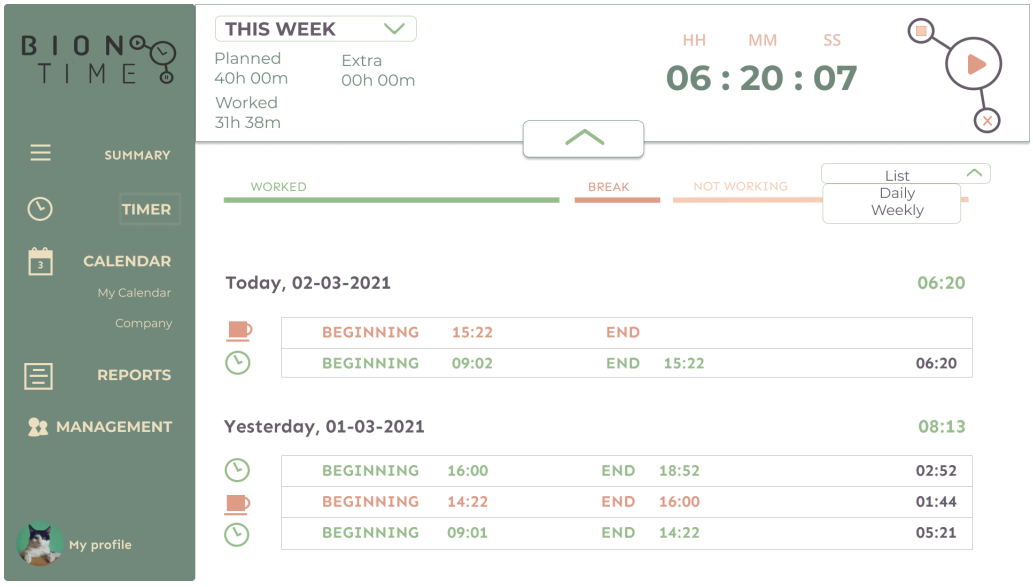


Figura 3.16: Vista de Timer List

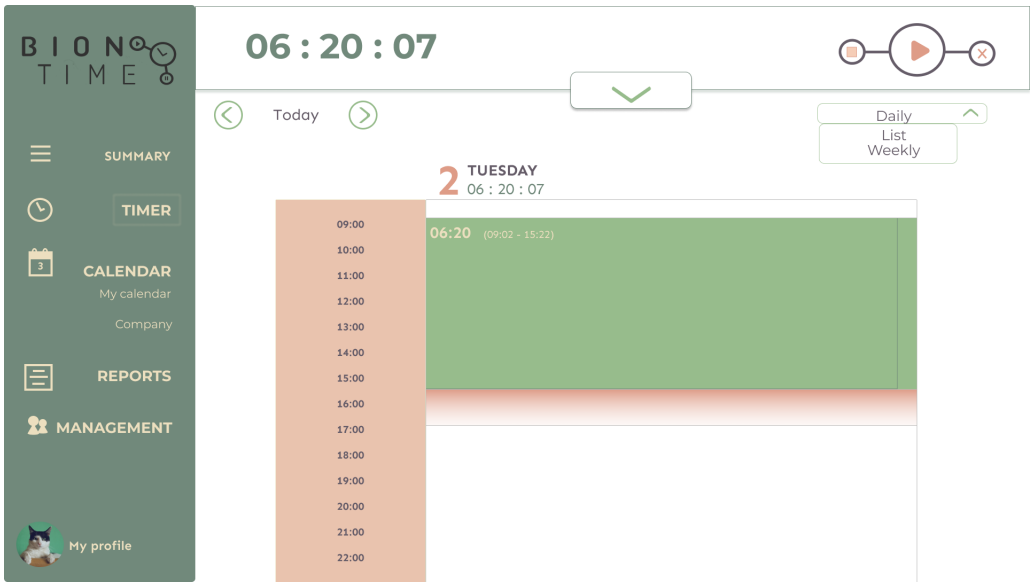


Figura 3.17: Vista de Timer Daily

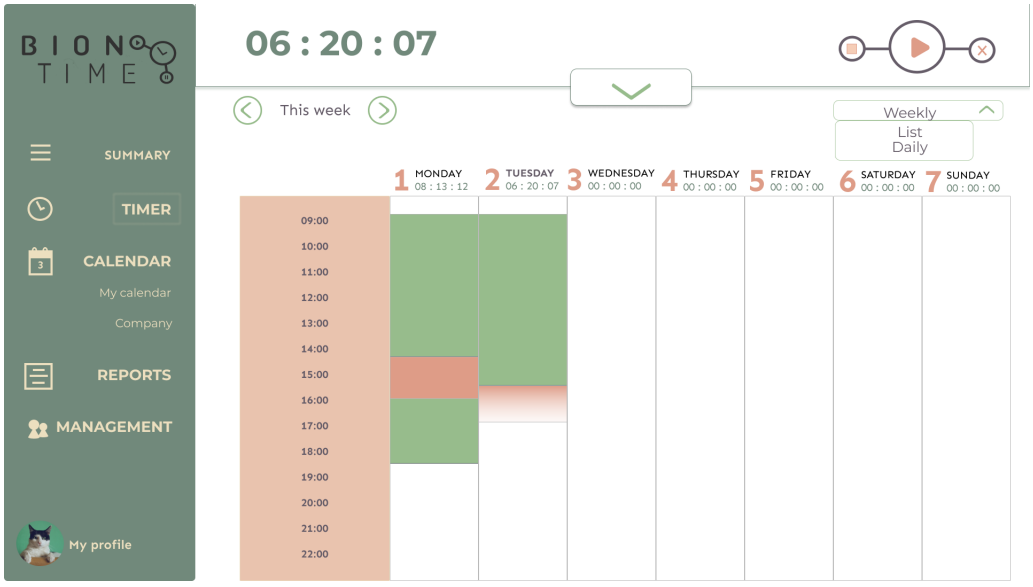


Figura 3.18: Vista de Timer Weekly

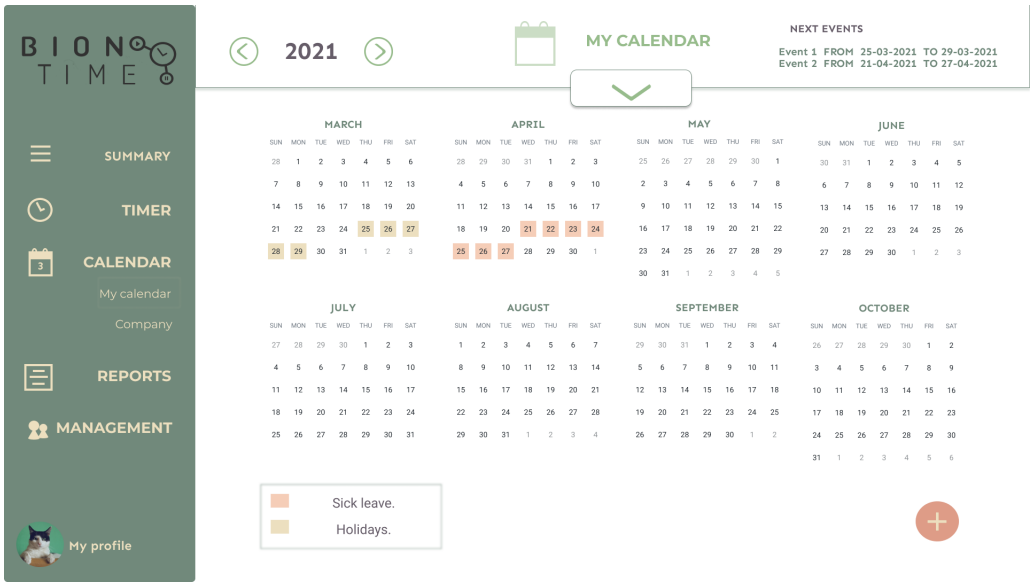


Figura 3.19: Vista de My calendar

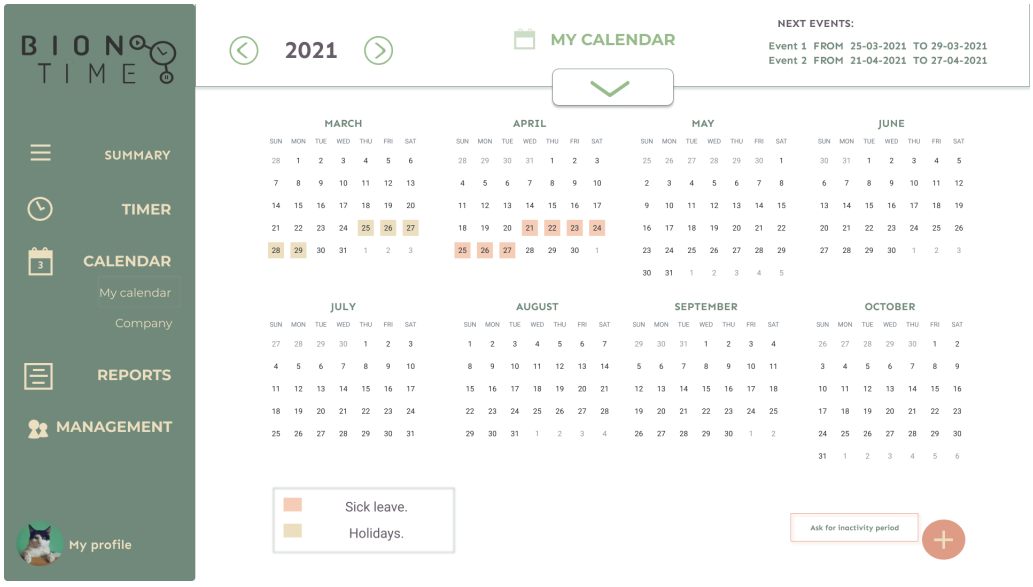


Figura 3.20: Vista de My calendar

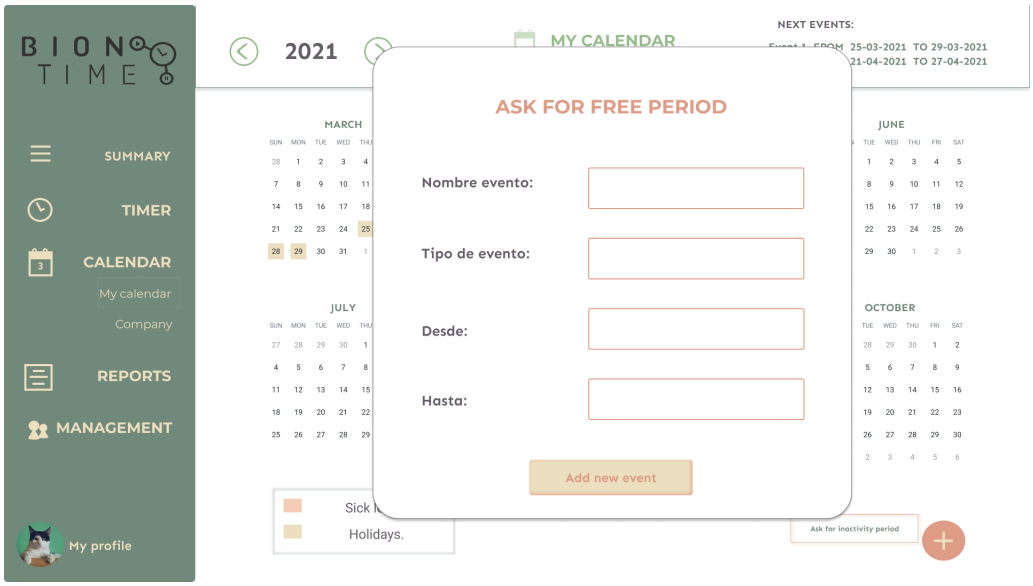


Figura 3.21: Vista de Add Event

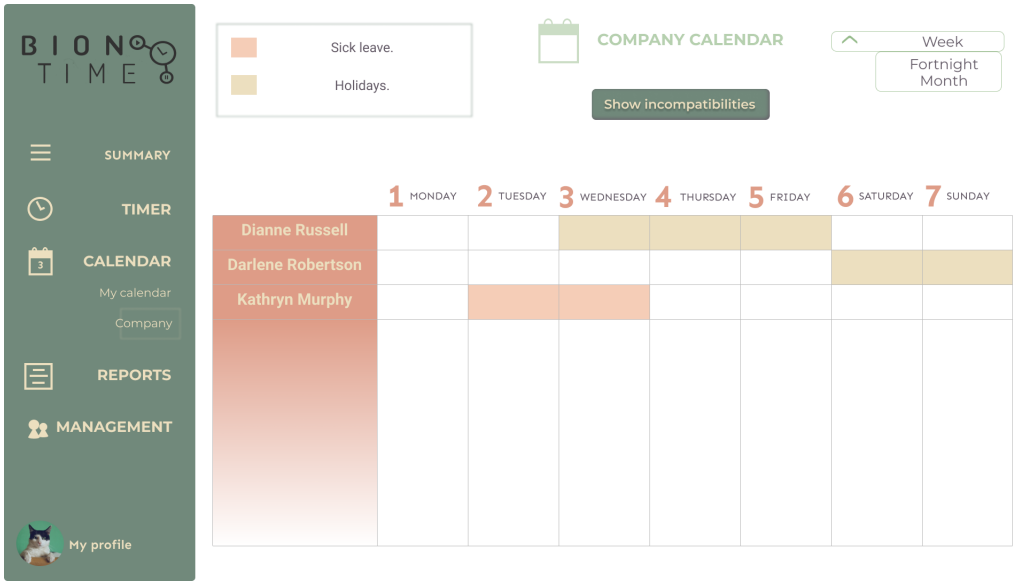


Figura 3.22: Vista de Company calendar

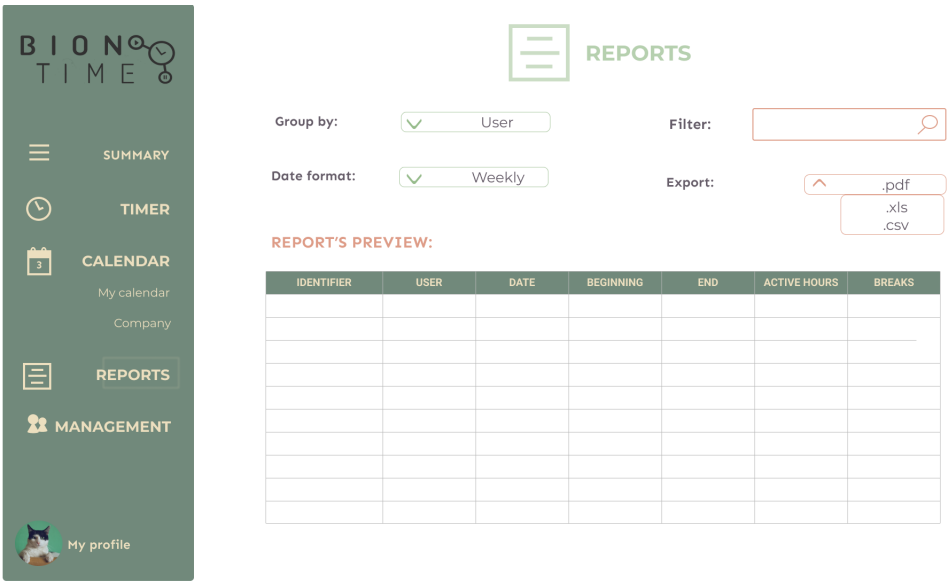


Figura 3.23: Vista de Reports

Figura 3.24: Vista de Management

Figura 3.25: Vista de Add user

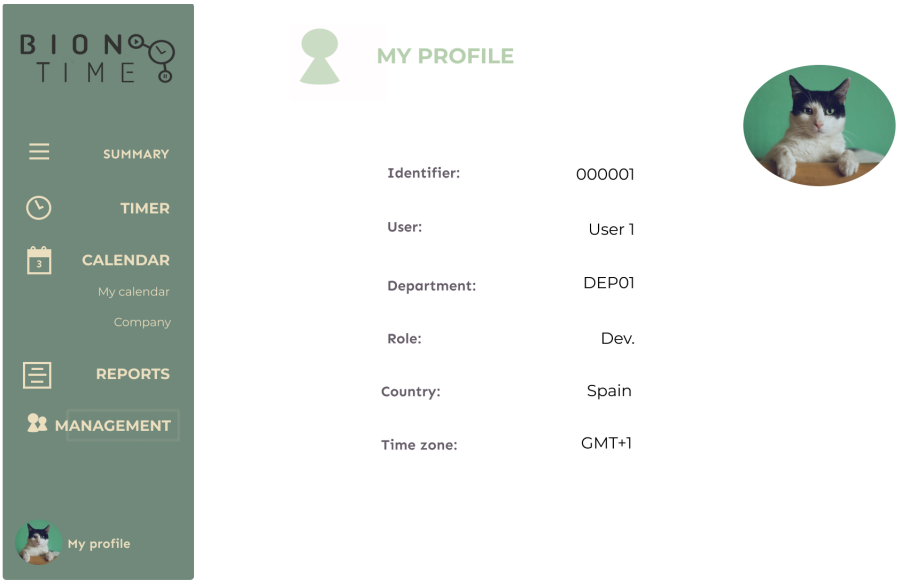


Figura 3.26: Vista de My profile

Capítulo 4

Implementación

4.1. Tecnologías utilizadas

Un **sitio web** [33] es un conjunto de páginas web relacionadas y con un dominio o subdominio en común dentro de Internet. A estas páginas se accede normalmente utilizando un URL raíz común. Las páginas web son archivos de código HTML (*Hyper Text Markup Language*) que conforman el sitio web.

El **desarrollo web** [34] consiste en crear sitios web. Para esto es necesario el uso de tecnologías *software* que suponen una combinación de procesos de gestión de datos y el uso de un navegador con el objetivo, bien de realizar ciertas tareas o de mostrar información.

A continuación se explican algunas de las tecnologías más importantes utilizadas para realizar este proyecto:

4.1.1. Laravel

Laravel es un *framework* para aplicaciones web, [35], utilizado ya por la empresa cliente; que proporciona una estructura y punto de partida para crear una aplicación, eliminando así las dificultades que supone empezar desde cero y permitiendo centrarse en los detalles que marcarán la diferencia.

Para poder crear un proyecto con Laravel, se brindan varias opciones, entre las cuales, la elegida requiere tener instalado PHP y *Composer* y ejecutar los siguientes comandos:

```
composer create-projet laravel/laravel <nombre-proyecto>

cd example-app

php artisan serve
```

Las actualizaciones del nuevo proyecto creado se mostrarían también en un repositorio de **GitHub** perteneciente a la empresa cliente. Para crear este nuevo repositorio sería suficiente con ejecutar el siguiente comando:

```
laravel new <nombre-proyecto> --github
```

Estructura de directorios [36]

El directorio *Root*: Dentro de este directorio se encuentran los siguientes:

- El **directorio *App***: Que contiene el código principal de la aplicación.
- El **directorio *Bootstrap***: que contiene el fichero *app.php* que arranca el *framework*. Este directorio también contiene un directorio *cache* con los ficheros generados por el *framework* para optimizar el rendimiento.
- El **directorio *Config***: Este directorio contiene todos los archivos de configuración de *Laravel*. Concretamente, contiene el fichero **.env** [37], donde se establecerá el sistema gestor de bases de datos a utilizar, *PostgreSQL*.
- El **directorio *Database***: que contiene las migraciones de la base datos, las factorías de los modelos y, en algunos casos, las semillas de datos.
- El **directorio *Public***: que contiene el fichero *index.php*, el punto de entrada de todas las peticiones a la aplicación a partir del cual se genera una instancia de la misma.
- El **directorio *Resources***: contiene las vistas, así como otros archivos de código, como las componentes Vue en este caso o los ficheros *sass*.
- El **directorio *Routes***: contiene todas las definiciones de las rutas para la aplicación. En concreto, el fichero *web.php* contiene la mayoría de rutas definidas para la aplicación.
- El **directorio *Storage***: contiene los logs, archivos caché, y otros ficheros generados por el *framework*. Contiene, entre otros, el directorio *storage/app/public*, utilizado para guardar archivos generados por el usuario, como las imágenes de usuario, que deben ser de acceso público.
- El **directorio *Tests***: contiene los tests automatizados, que se ejecutan utilizando uno de los tres comandos:

  ```
  phpunit    php vendor/bin/phpunit    php artisan test
  ```

  Este último muestra una representación más detallada de la ejecución de los tests.
- El **directorio *Vendor***: Que contiene las dependencias de *Composer*.

El directorio *App*: Dentro de este directorio se encuentra la mayor parte de la aplicación. Contiene otros directorios como *Console*, *Http* y *Providers*.

- El **directorio *Console***: contiene los comandos *Artisan*.
- El **directorio *Http***: contiene los controladores, *middleware* y las peticiones y el archivo *Kernel.php* que es el encargado de establecer la configuración y definir la lista de *middleware* HTTP por el que deben pasar las peticiones que serán filtradas y examinadas antes de entrar en la aplicación.
- El **directorio *Models***: contiene todas las clases correspondientes al modelo *Eloquent* asociado a la base de datos. Cada tabla de la base de datos tiene un modelo correspondiente usando para interactuar con la tabla. Estos permiten solicitar datos a las tablas, así como insertar nuevos registros en ellas.
- El **directorio *Providers***: contiene todos los proveedores de servicio de la aplicación, encargados de arrancar los diversos componentes del *framework*, como la base de datos o las rutas, entre otros.

Enrutado

Las rutas [38] más básicas en Laravel aceptan una URI en una sintaxis similar a la siguiente:

```
use Iluminate\Support\Facades\Route;

Route::get('/ruta', function () {
return 'Hello World';
});
```

La declaración de la función puede sustituirse por la llamada a una función de otra clase, normalmente definida en un controlador, de la siguiente forma:

```
use App\Http\Controllers\SomeController;

Route::get('/ruta', [SomeController::class], 'funcion']);
```

Una sintaxis parecida, puede utilizarse para insertar, actualizar o eliminar registros de la base de datos:

```
use App\Http\Controllers\SomeController;

Route::post('/ruta', [SomeController::class], 'funcion']);
```

Cualquier argumento enviado a la función en el momento de la llamada se recibe utilizando la clase *Request*, bien con la sintaxis:

```
use Iluminate\Http\Request;
Route::get('/ruta', function (Request $request) {
});
```

o bien definiendo el argumento del mismo modo en el controlador donde esté definida la función.

Todas las rutas definidas para la aplicación se localizan en el directorio *routes*. Concretamente, las correspondientes a las interfaces web de la aplicación se encuentran en el archivo *routes/web.php*.

Es posible definir grupos de rutas utilizando el método *middleware* con una sintaxis similar a la siguiente:

```
Route::middleware('grupo')->group(function (){
    Route::get('/ruta1', function () {

    });

    Route::get('/ruta2', function () {
    });

});
```

Vistas

Las vistas [39] se encargan de separar la lógica de los controladores del código HTML y se localizan en el directorio *resources/views*. La sintaxis es similar a la siguiente:

```
<html>
   <body>
      // Contenido
   </body>
</html>
```

La parte correspondiente al contenido es diferente al utilizar *Laravel* con *Vue*, ya que en lugar de utilizar simples elementos HTML, estos son sustituidos por componentes Vue, como se explicará más adelante.

Estas vistas se almacenan en ficheros con un nombre similar a: *resources/views/vista.blade.php*. Para obtener estas vistas, las funciones ejecutadas por la ruta utilizarán esta sintaxis:

```
function() {
   return view('vista', ['datos' => datos]);
});
```

Eloquent

Laravel incluye un mapeo objeto-relacional u ORM que facilita la interacción con la base de datos [40]. Cada tabla de la base de datos se corresponde con un "Modelo" utilizado para interactuar con dicha tabla, permitiendo insertar, actualizar y eliminar registros de la misma, además de simplemente mostrarlos.

Estos Modelos se almacenan en el directorio *app/Models* y heredan de la clase *Illuminate/Database/Eloquent/Model*. Estos pueden generarse utilizando el siguiente comando:

```
php artisan make:model ModeloEjemplo
```

A menos que se especifique a qué tabla corresponde dicho modelo, *Eloquent* asumirá que esta es *modelo_ejemplos*. Para indicar a qué tabla debe asociarse el modelo se utiliza la siguiente asignación:

```
protected $table = 'tabla_del_modelo';
```

Una línea parecida se utiliza para definir la clave primaria del modelo:

```
protected $primaryKey = 'id_del_modelo';
```

Eloquent asumirá a menos que se le indique lo contrario, que esta clave primaria es autoincremental. En cuanto a las claves primarias compuestas, *Eloquent* no da soporte a este tipo de claves, necesita al menos un único identificador que sirva como *primary key*.

Para acceder a los registros de la tabla correspondiente al modelo, Eloquent proporciona una serie de métodos que generan resultados similares a los obtenidos con las sentencias SQL. Por ejemplo la llamada

`ModeloEjemplo::all()`

sería similar a:

```
SELECT * FROM tabla_del_modelo;
```

Otros métodos serían: *where('columna', valor), orderBy('columna'), take(número)* o *get()*. Así, por ejemplo, similar a la sentencia SQL:

```
SELECT columna2 FROM tabla_del_modelo WHERE columna=valor;
```

Sería lo siguiente:

```
$registros = ModeloEjemplo::where('columna', valor)->get();
foreach($registros as $reg){
     echo $reg->columna2;
}
```

Si se quiere acceder solo a un registro, se pueden usar los métodos *find, first* o *firstWhere* como se muestra a continuación:

```
$registro = ModeloEjemplo::find(1);
$registro = ModeloEjemplo::where('columna', valor)->first();
$registro = ModeloEjemplo::firstWhere('columna', valor);
```

Insertar o actualizar registros: Para **insertar** un registro, con datos, por ejemplo, provenientes de una petición desde la vista se crea una nueva instancia del modelo y se fijan sus atributos. Después, se guarda utilizando el método *save* con la siguiente sintaxis:

```
public function insertar(Request $request){
    $modelo = new ModeloEjemplo;
    $modelo->columna = $request->valor_columna;
    $modelo->save();
}
```

Al llamar al método *save* se crea un nuevo registro en la tabla de la base de datos correspondiente al modelo y con los datos indicados en la asignación de atributos.

Si lo que se quiere es **actualizar** un registro, se realiza una consulta como las indicadas anteriormente sobre un solo registro y se asigna un nuevo valor a sus atributos. Una vez modificados los valores deseados se llama al método *save* del mismo modo que se hace con las inserciones.

Eliminar registros: Para eliminar un registro, al igual que para actualizarlo, se busca este registro y, en lugar de utilizar el método *save* se llama al método *delete*. Si se quieren eliminar todos los registros del modelo se utiliza el método *truncate*. Utilizando el método *destroy*, se puede eliminar un registro si se conoce su clave primary, pasándola como argumento.

4.1.2. Vue

Vue [41] [42] [43] es un *framework* progresivo para construir interfaces de usuario. A diferencia de otros *frameworks*, está diseñado para aprenderse de forma incremental. La librería principal se centra únicamente en la capa de la vista y se puede integrar con otras librerías o proyectos existentes.
La plantilla principal de Vue tiene una sintaxis basada en plantillas de HTML unido a código en *javascript* o, en en el caso de este proyecto, *typescript*:

```
<div id="app">
    {{ datos }}
</div>
```

La sintaxis utilizada para mostrar los datos dentro del "div" se conoce como "Mustache" ({{ }}):

```
var app = new Vue({
    el: '#app',
    data: {
        datos: 'Valor de datos'
    }
})
```

Estos datos se pueden asignar también a propiedades del HTML utilizando una directiva de Vue, *v-bind*, como se muestra a continuación:

```
<div id ="app">
    <span v-bind:title="datos">
    </span>
</div>
```

Las directivas de Vue están precedidas por "v-". Otras directivas útiles son *v-if* y *v-else*. De tal forma que si se les pasa una expresión booleana muestren un contenido u otro con una sintaxis similar a la siguiente:

```
<div id="app">
    <span v-if="variable_bool">Visible si variable_bool es True</span>
    <span v-else>Visible si es False</span>
```

Otra directiva importante es *v-for*, que puede utilizarse para mostrar los elementos de una lista o para rellenar, por ejemplo, una tabla:

```
<div id="app">
   <ol>
      <li v-for="el in elementos">
        {{ el.value }}
      </li>
   </ol>
</div>
```

Para asociar funciones o código a un evento, por ejemplo a pulsar un botón, se utiliza una sintaxis como la siguiente:

```
<div id="app">
   <button @click="funcionClick"> Pulsa aquí </button>
</div>
```

Si se desea, por ejemplo, que un valor introducido se asocie a su vez a una variable se puede utilizar la directiva *v-model*:

```
<div id="app">
   <input v-model="valor">
</div>
```

Componentes

El sistema de componentes [41] es uno de los conceptos más importantes de este *framework* ya que permite construir una aplicación grande de forma modular, dividiendo su interfaz en partes más pequeñas que pueden incluso reutilizarse.
La sintaxis para definir una componente es similar a la siguiente:

```
Vue.component('componente', {})
var app = new Vue(...)
```

En dicho código se crea también una instancia de Vue, que será la instancia raíz y que contendrá otras componentes o instancias.

Una vez creada una componente, se puede utilizar dentro de otros elementos HTML y recibir datos desde el elemento padre con la sintaxis ":" + "nombre de la propiedad", en este caso de forma dinámica:

```
<div>
    <componente :propiedad="var"></componente>
</div>
```

Esta sintaxis ":- "nombre", es una abreviación de *v-bind:* y se puede utilizar también para asignar valores a las propiedades propias de los elementos de HTML, como *id*, *class*, *disabled*, etc., cuya asignación puede además, depender de alguna condición. Por ejemplo:

```
<div>
    <componente :class="{nombre_clase: condicion}"></componente>
</div>
```

Así, solo si la condición se cumple, se asignará a la clase de la componente el valor *nombre_clase*. Esto puede volverse más complejo si, por ejemplo, se quiere asignar una lista de valores a la propiedad o si se quiere asignar uno u otro valor dependiendo de una condición.

También existe la posibilidad de indicar a una función que actualice propiedades de la componente utilizando la sintaxis "@Watch('valor')" siempre que la variable "valor" cambie.

Para utilizar una componente dentro de otra se importa indicando su ruta relativa:

```
import ComponenteA from './ComponenteA.vue'
```

Filtros

Vue te permite utilizar lo que se conoce como filtros, que pueden servir, por ejemplo, para formatear el contenido de un campo de texto. La sintaxis utilizada para aplicarlos a una variable o valor es la siguiente:

```
{{ variable | filtro }}
```

Donde el filtro es una función definida en la sección de filtros de la componente. Pueden además, encadenarse, aplicando otro filtro a lo obtenido con el anterior y a su vez pueden llevar también argumentos, como se muestra a continuación:

```
<div :class="variable | filtro1 | filtro2(argumentos) "></div>
```

4.1.3. PostgreSQL

Una **base de datos** [44] [45] es una colección organizada de datos utilizada para modelar aspectos relevantes de la realidad. Para gestionar estos datos, es decir: almacenarlos, extraerlos, modificarlos, eliminarlos, etc. se hace uso de sistemas software conocido como gestores de bases de datos. Estos permiten manejar los datos de forma segura, garantizando las restricciones de integridad y minimizando las inconsistencias.

Una **base de datos relacional** [46] representa un conjunto de relaciones. Para la manipulación de estas bases de datos es necesario un lenguaje de consulta o *Query Language*. El lenguaje **SQL** o *Structured Query Language* [47] es el utilizado para gestionar este tipo de bases de datos. Basado en el álgebra y el cálculo relacionales, este lenguaje es el que utiliza el gestor ***PostgreSQL***, con el que se manejarán los datos en este proyecto.

PostgreSQL [48] [49] es un gestor de bases de datos relacionales de código abierto y orientado a objetos que se encuentra entre los más potentes del mercado. Como se puede intuir por su nombre, hace uso del lenguaje SQL.

Entre las ventajas de este gestor se encuentran el acceso gratuito al mismo; su disponibilidad, es compatible con múltiples plataformas y la mayoría de sistemas operativos actuales; su estabilidad y escalabilidad, lleva más de 20 años en el mercado y permite adaptar su configuración a los recursos *hardware* de los que disponga el usuario; así como su robustez y fiabilidad, dado que sigue el protocolo ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Implementa además la mayoría de funcionalidades del estándar SQL.

4.2. Entorno de desarrollo

Para llevar a cabo este trabajo se han utilizado dos editores de código fuente:

Principalmente se ha utilizado **Visual Studio Code (VSCode)** [1] que permite su integración con **Git**, además de numerosas extensiones que facilitan el desarrollo de *software* [50].

El otro editor de código, menos utilizado que el anterior, ha sido **PhpStorm** [2], destinado principalmente a la visualización del modelo de la base de datos y en ocasiones a la edición de los modelos asociados al mismo.

[1] `https://code.visualstudio.com/`

[2] `https://www.jetbrains.com/es-es/phpstorm/`

4.3. Lenguajes

Los lenguajes utilizados para llevar a cabo este proyecto son:

- **PHP *Hypertext Preprocessor*** [51], un lenguaje interpretado de código abierto, que se encuentra entre los más utilizados en desarrollo web y puede ser incrustado en HTML y que está centrado en la programación de *scripts* del lado del servidor.
- **Typescript** [52], otro lenguaje de código abierto que extiende la sintaxis de JavaScript y añade a esta el uso de tipos estáticos y objetos basados en clases.
- Además de estos, se utilizan **HTML *(HyperText Markup Language)*** [53] y **SASS *(Syntactically Awesome StyleSheets)*** [54]. El primero para definir el significado y la estructura de la web, mientras que el segundo, un metalenguaje de diseño gráfico u hojas de estilo en cascada basado en CSS (*Cascading Style Sheets*), define la apariencia o presentación de la misma.

La combinación de estos lenguajes presta la funcionalidad, estructura y apariencia necesarias para que la aplicación web pueda proporcionar los servicios que se esperan de la misma.

4.4. Otras herramientas

Además de los lenguajes, *frameworks* y entornos de desarrollo utilizados, han resultado útiles para la planificación y el diseño, dos herramientas:

- **Trello** [3], un *software* para la administración de proyectos de gran flexibilidad que permite organizar y priorizar actividades a usuarios individuales o a equipos. Se basa en el uso de tarjetas, listas y tableros, a los que se pueden añadir *plugins*, etiquetas u otros elementos que permitan adaptar su uso a los requisitos de cada usuario [55].
- **Figma** [4], una aplicación web utilizada para el prototipado y pensada especialmente para el diseño de interfaces y experiencia de usuario [56].

También, para el control de versiones, se ha utilizado **GitHub** [5], una de las principales plataformas de desarrollo colaborativo creada para alojar el código de las aplicaciones y que hace uso del *software* de control de versiones Git [57] [58].

Concretamente, se ha mantenido una rama principal, *main*, a la que se subiría todo el desarrollo una vez probado y otra rama secundaria, *dev*, a la que se iban incorporando las actualizaciones de cada *epic*.

[3] `https://trello.com/es`
[4] `https://www.figma.com/`
[5] `https://github.com/`

Estas actualizaciones se desarrollaban a su vez en una rama aparte con el nombre *in#Nep#Mnombre*. En esta sintaxis, #N representa el número de la *initiative* correspondiente y #M el número de la *epic*. En cuanto a *nombre*, este valor hace alusión a un nombre descriptivo de la temática de la *epic* a la que está asociada dicha rama como se muestra de forma genérica en la **Figura 4.1**.

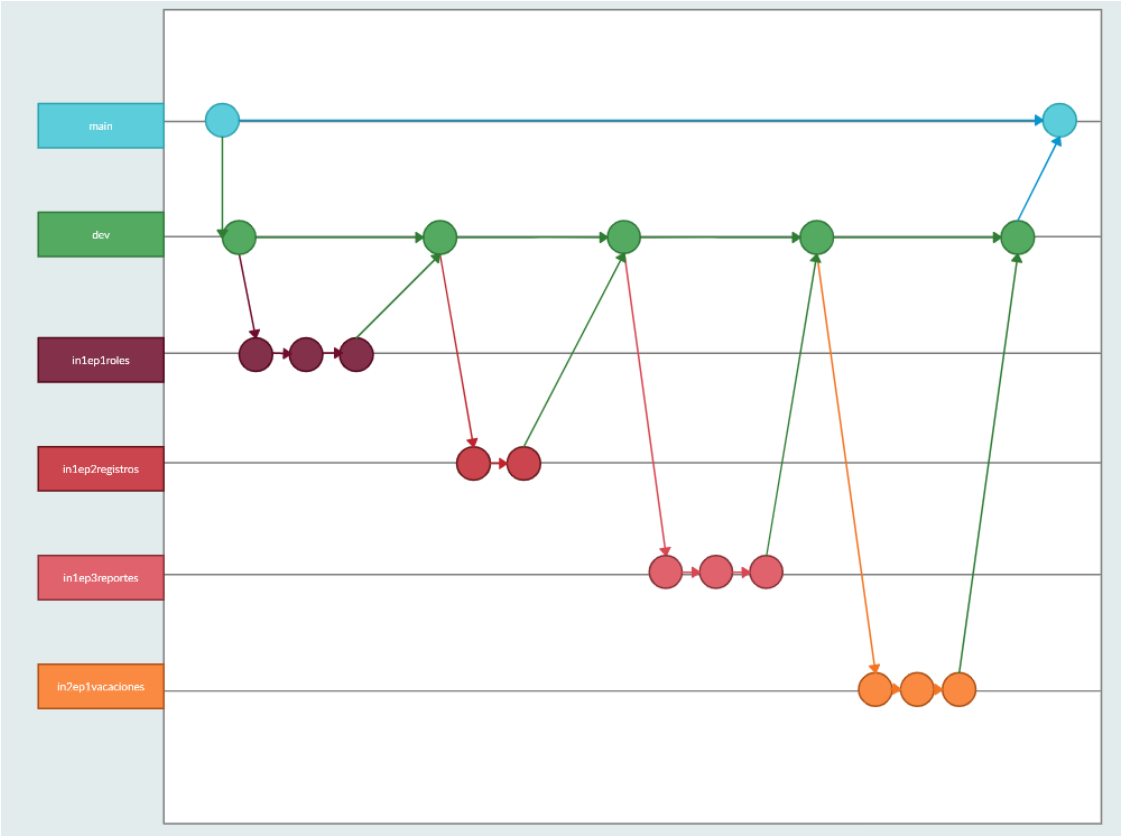


Figura 4.1: Control de versiones

4.5. Desarrollo

4.5.1. Desarrollo *Front End*

El **desarrollo *Front End*** se basa en las vistas de *Laravel* con la extensión *blade.php* y las componentes *Vue* que conforman el contenido de las mismas. Para ejemplificar esta parte de la implementación se utilizará la vista correspondiente al *Dashboard* o *Summary*, el resumen que encuentra el usuario al entrar a la aplicación.

En primer lugar, todas las vistas se crean a partir de un fichero con la extensión *blade.php*, como el que se muestra en la **Figura 4.2**, situado en la ruta *resources/views/dashboard.blade.php*. En esta puede verse cómo el *dashboard* es un elemento hijo del único elemento padre con id *app* y que para su creación recibe como información los datos con formato JSON: *raw_user, hours, percentages, laststate* y *events*. Estos se envían a la vista desde el controlador del fichero *RecordController.php*, como se explicará en la sección correspondiente al **desarrollo *Back End***.

dashboard.blade.php ×

resources > views > dashboard.blade.php

```
1  @extends('base')
2
3  @section('content')
4      <div id="app">
5          <dashboard
6          :raw_user=@jsvue($user)
7          :hours=@jsvue($hours)
8          :percentages=@jsvue($percentages)
9          :laststate=@jsvue($laststate)
10         :events=@jsvue($events)></dashboard>
11     </div>
12 @endsection
13
14
15
```

Figura 4.2: Fichero *dashboard.blade.php*

Como puede verse en la línea 1 del fichero *dashboard.blade.php* este extiende del fichero *base.blade.php*, en la **Figura 4.3**, que contiene la estructura básica HTML.

Figura 4.3: Fichero *base.blade.php*

La componente correspondiente al *Dashboard* se encuentra en la ruta *resources/js/components/Dashboard.vue*. En la **Figura 4.4** se puede ver su estructura, prácticamente igual a la de la componente de *Timer* y muy similar a la del resto de componentes. En concreto, se compone de tres partes, una primera, común a todas las secciones, la componente *LeftBar* que se mostrará después en más detalle; una segunda correspondiente a la barra superior donde se crean los registros en la vista de *Timer* y la de *Dashboard* y por último el contenido de la sección. En este caso, el contenido se encuentra dentro del *div* etiquetado como *dashboard__content bmk__view__content*. Dentro de este se localizan los dos gráficos circulares, así como la tabla de eventos, también creada dentro de otroa componente y el link para solicitar que se muestre una guía sobre la aplicación.

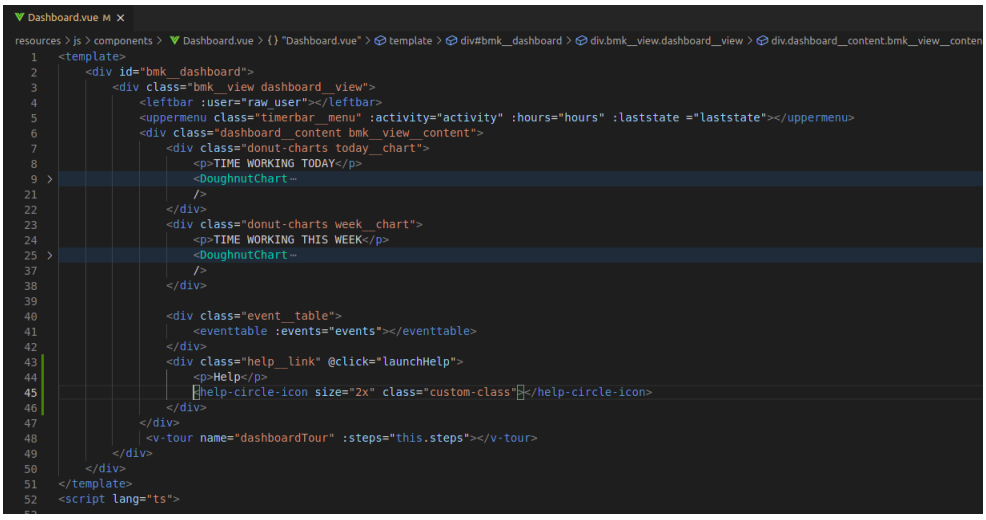


Figura 4.4: Fichero *Dashboard.vue*

Para poder utilizar componentes o elementos externos al proyecto es necesario importarlos, como ocurre con las componentes *Leftbar*, *UpperMenu* o *EventTable*. Lo mismo es necesario para utilizar los iconos como *HelpCircleIcon* o los gráficos de donut o *DoughnutChart*.

```
54  import {Component, Prop, Vue, Watch} from 'vue-property-decorator';
55
56  // Models
57  import {ActivityList} from '../models/ActivityList';
58  import User from '../models/User';
59
60
61  //Components
62  import Leftbar from './UIBasics/Leftbar.vue';
63  import UpperMenu from './UIBasics/UpperMenu.vue';
64  import EventTable from './UIBasics/EventTable.vue';
65
66  // Icons
67  import {HelpCircleIcon } from 'vue-feather-icons'
68
69
70
71  // @ts-ignore
72  import DoughnutChart from 'vue-doughnut-chart/src/DoughnutChart.vue';
73
```

Figura 4.5: Fichero *Dashboard.vue*

Después de importarlos, estos elementos se declaran como componentes de la componente, en este caso de *Dashboard*. Además, en la creación de la clase correspondiente, se reciben las propiedades enviadas desde el controlador al crearla como puede verse en la **Figura 4.6**. También se pueden declarar variables que no se reciban como propiedades, como se hace con los pasos, la variable *steps*, correspondientes al tutorial de la aplicación.

```
76  @Component({
77      components: {
78          HelpCircleIcon,
79
80          Leftbar, UpperMenu,
81
82          DoughnutChart,
83
84          EventTable
85      }
86  })
87  export default class Dashboard extends Vue {
88
89
90      @Prop(Object)
91      raw_user!: {};
92
93      @Prop()
94      hours!: any[];
95
96      @Prop()
97      percentages!: any[];
98
99      @Prop()
100     laststate!: string;
101
102     @Prop()
103     events!: any[];
104
105
106     // Steps for help guide
107 >   steps = […
169     ]
```

Figura 4.6: Fichero *Dashboard.vue*

Por último, se pueden declarar funciones como la que se muestra en la **Figura 4.7**, *launchHelp*, llamada al pulsar sobre el icono de ayuda y que muestra el tutorial sobre la página del *Dashboard*.

```
178     // object activityList for timer view
179     activity: ActivityList = new ActivityList();
180
181
182
183     /* Donut Chart */
184
185     bgcolor: string = "#DADADA"
186     tdcolor: string = "#71897b"
187     wkcolor: string = "#DE9C87"
188
189     todaypercent: number = (this.percentages[0]['today'] > 100.0) ? 100 : this.percentages[0]['today'].toFixed(2);
190     weekpercent: number = (this.percentages[0]['week'] > 100.0) ? 100 : this.percentages[0]['week'].toFixed(2);
191
192
193     @Watch('activity.records')
194     reload(){
195         window.location.href = "/dashboard";
196     }
197
198     launchHelp(){
199         this.$tours['dashboardTour'].start();
200     }
201
202     async mounted() {
203         console.log("dashboard mounted executed");
204         await this.activity.reload();
205
206     }
207
208 }
209
210 </script>
```

Figura 4.7: Fichero *Dashboard.vue*

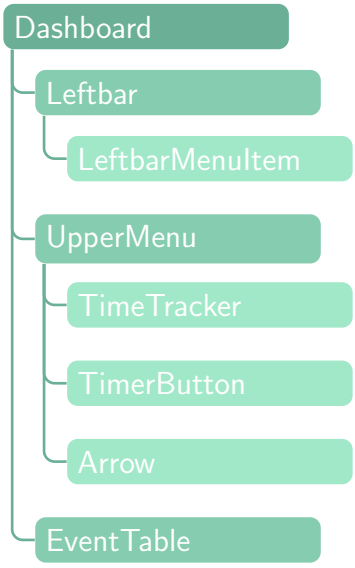
Como ejemplo de otra componente, en este caso común a todas las demás y localizada en la ruta *resources/componentes/UIBasics/Leftbar.vue*, se muestra la estructura de la componente *Leftbar* en la **Figura 4.8**. Esta componente ejemplifica cómo ayudan a encapsular partes de un elemento, de forma que este, que podría ser en un principio difícil de comprender, pase a tener unas dimensiones aceptables, siendo así más sencillo y permitiendo a su vez repetir elementos sin necesidad de añadirlos por completo cada vez que se defina un elemento que los contenga.

```
Leftbar.vue ×
resources > js > components > UIBasics > Leftbar.vue > {} "Leftbar.vue" > template > aside.leftbar__menu > div.options__container > ul.leftMenu > li.menuItem > le
1   <template>
2       <aside class="leftbar__menu">
3           <div class="logo__img__container">
4               <img id="logoImg" src="/images/logoBION2.png">
5           </div>
6
7           <div class="options__container">
8               <ul class="leftMenu">
9                   <li v-for="(item, i) in items" :key="i" class="menuItem">
10                      <leftbaritem :link="item.route" :name="item.name" :id="item.name">
11                          <menu-icon v-if="item.name=='summary'" size="1.5x" class="menuIcon"></menu-icon>
12                          <clock-icon v-if="item.name=='timer'" size="1.5x" class="menuIcon"></clock-icon>
13                          <calendar-icon v-if="item.name=='calendar'" size="1.5x" class="menuIcon"></calendar-icon>
14                          <bar-chart-2-icon v-if="item.name=='reports'" size="1.5x" class="menuIcon"></bar-chart-2-icon>
15                          <users-icon v-if="item.name=='management'" size="1.5x" class="menuIcon"></users-icon>
16                          <div class="calendar__items" v-if="item.name=='calendar'">
17                              <a class="calendar__item" :href="'/calendar'">My calendar</a>
18                              <a class="calendar__item" :href="'/ccalendar'">Company</a>
19                          </div>
20                      </leftbaritem>
21                  </li>
22              </ul>
23          </div>
24          <div class="menuItem myProfile">
25              <a :href="'/myprofile'" class="profile">my profile</a>
26              <img :src='this.src' class="profile__image">
27          </div>
28          <v-tour name="leftbarTour" :steps="this.steps"></v-tour>
29      </aside>
30  </template>
31
```

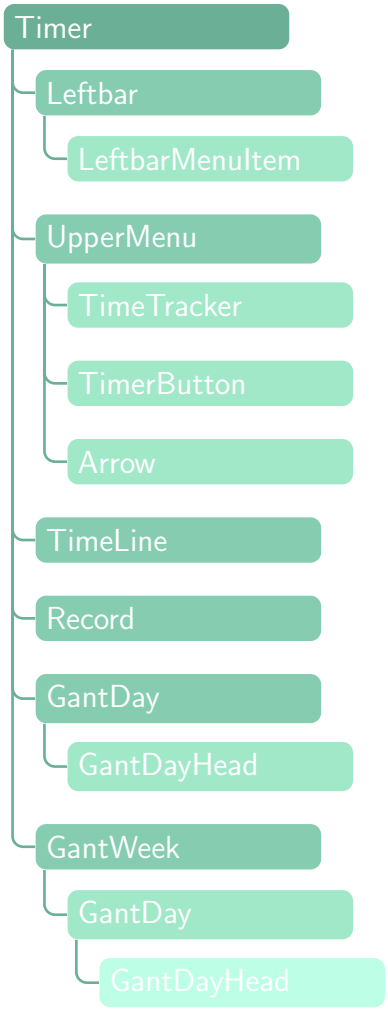
Figura 4.8: *Leftbar.vue*

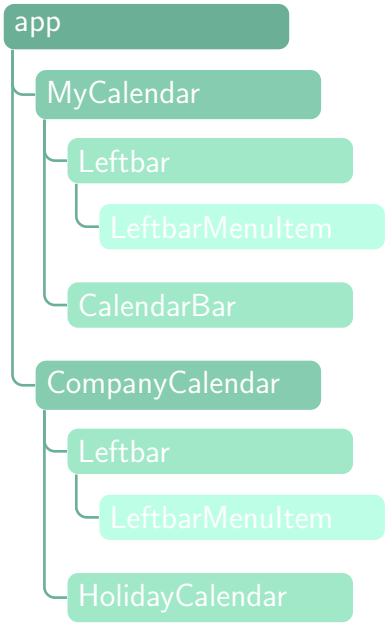
Además de las citadas componentes, la aplicación web de este proyecto está formada por otras muchas. En concreto, el árbol de componentes *Vue* que conforma la aplicación es el siguiente:

Separando por componentes para que no resulte demasiado complejo, el *Dashboard* presentaría el siguiente árbol de componentes:

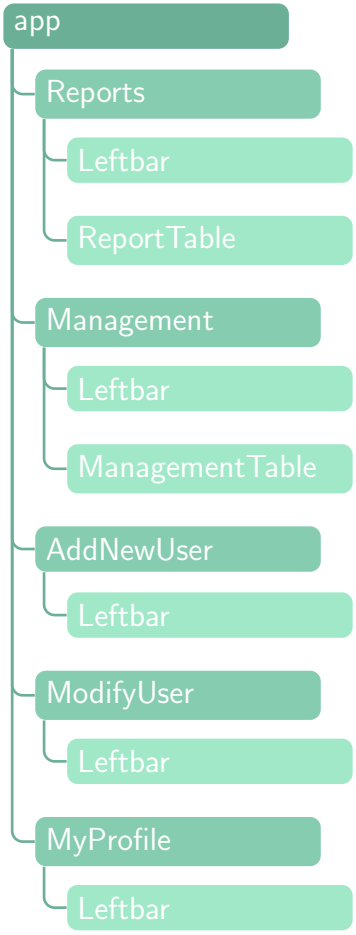


Y los de *Timer*, *MyCalendar* y *CompanyCalendar*





Por último, el resto de componentes, sin detallar la composición de *Leftbar*, que ya se ha indicado en las anteriores, tendrían el siguiente árbol:



4.5.2. Desarrollo *Back End*

Para traer la información desde la base de datos a las vistas se utilizan los controladores y modelos. Para seguir con el ejemplo del *Dashboard* se muestra en las **Figuras 4.9** y **4.10** cómo se extrae la información de los eventos desde el controlador *RecordController.php*, localizado en la ruta *app/Http/Controllers/Records/RecordController.php*.

En la primera imagen, se muestra la función *getDashboardInfo*. En esta, se obtiene la información del usuario que ha iniciado sesión, sus horas trabajadas, el estado de su último registro y los eventos del usuario. Además, se calculan los porcentajes de tiempo trabajado a partir de las horas obtenidas con la función *getWorked*, mediante la función *getPercentages*.
Una vez obtenida esta información se devuelve la vista junto con los datos que conformarán las propiedades de la componente correspondiente.

RecordController.php

app > Http > Controllers > Records > RecordController.php

```php
    ////// FOR DASHBOARD VIEW //////

    /**
     * Function to get UpperMenu Hours for logged user
     *
     * gets the hours worked (call to getWorked)
     *
     * gets percentages of hours worked based on hours obtained (getPercentages)
     * Percentage of time working today,
     * Percentage of time working this week,
     *
     * gets the state of the last record (call to getLastRecordState)
     *
     * gets events of user logged
     *
     * sends this info to launch Dashboard View
     */
    public function getDashboardInfo(){

        $user = Auth::user();

        // UpperMenu Hours
        $hours = $this->getWorked();

        //Percentages
        $percentages = $this->getPercentages($hours);

        // State of the last record
        $laststate = $this->getLastRecordState();

        // Events of user
        $events = $this->getEvents();


        return view('dashboard', [
            'user' => $user,
            'hours' => $hours,
            'percentages' => $percentages,
            'laststate' => $laststate,
            'events' => $events,
        ]);

    }
```

Figura 4.9: Función *getDashboardInfo*

La segunda imagen muestra una de estas funciones llamadas desde *getDashboardInfo*, en concreto *getEvents* que utiliza el modelo *Holiday* para extraer aquellos eventos asociados al usuario con *user_id* el del usuario loggeado, ordenados por *id*.

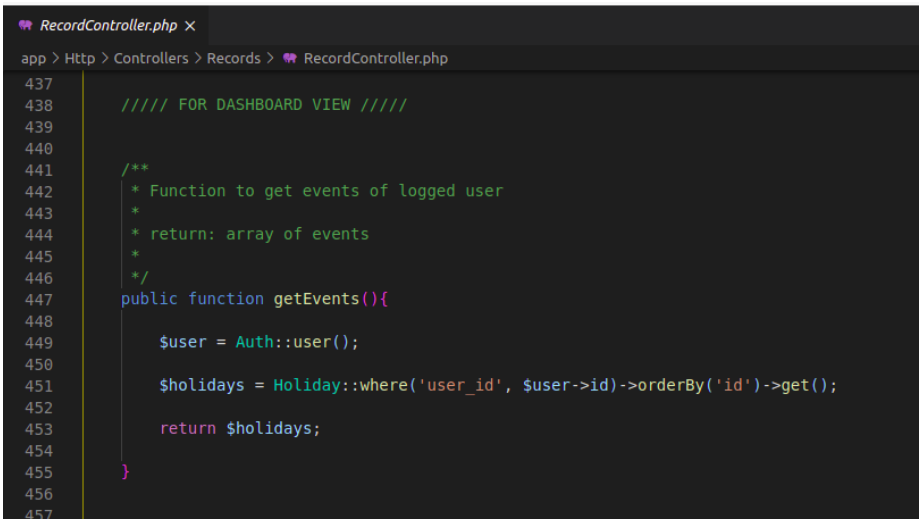


Figura 4.10: Función *getEvents*

El modelo utilizado en la función *getEvents*, *Holiday*, se encuentra en la ruta *app/Http/Models/Holiday.php* y se muestra en la **Figura 4.11**. Este se asocia a la tabla *holidays* de la base de datos y tiene como *primary key* la columna *id*. La función *user()* devuelve los ids de los usuarios a los que corresponde cada evento.

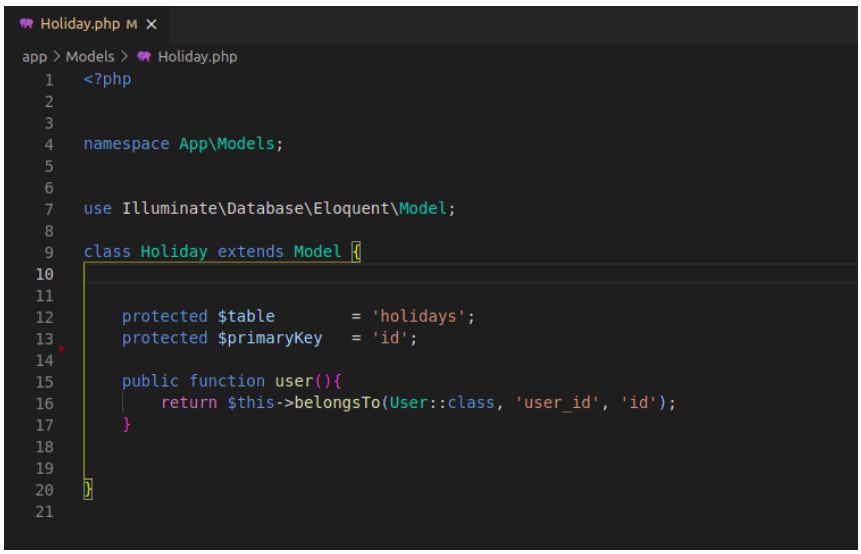


Figura 4.11: Modelo *Holiday*

Implementación a nivel de base de datos

Para definir la base de datos, *Laravel* proporciona las conocidas como *migrations*. Las básicas utilizadas en este proyecto definen dos funciones, una primera *up* para la creación de tablas, modificaciones e inserciones en la base de datos y otra segunda *down*, para la eliminación de tablas.
En estas funciones se añade el código SQL mediante la sintaxis

```
DB::statement("código SQL");
```

Un ejemplo de esto se muestra en las **Figuras 4.12** y **4.13**. Ambos ficheros se encuentran en la ruta *databas/migrations*. El primero incluye, entre otras sentencias, la creación de la tabla *holidays* y en la función *down* las sentencias para eliminar esta tabla y las tablas *holidays_limits* y *companies*, también creadas en este fichero.

2021_04_27_122635_init_holidays.php ×

database > migrations > 2021_04_27_122635_init_holidays.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use Illuminate\Support\Facades\DB;

class InitHolidays extends Migration
{
    /** …
    public function up(){
        DB::statement("…
        ");
        DB::statement("…
        ");

        DB::statement("
            CREATE TABLE holidays(
                id          serial PRIMARY KEY,
                user_id     integer NOT NULL REFERENCES users(id) ON UPDATE CASCADE ON DELETE RESTRICT,
                staring_date TIMESTAMP(0) NOT NULL,
                ending_date  TIMESTAMP(0) NOT NULL,
                created_at   TIMESTAMP(0) DEFAULT (CURRENT_TIMESTAMP AT TIME ZONE 'UTC'),
                updated_at   TIMESTAMP(0) DEFAULT (CURRENT_TIMESTAMP AT TIME ZONE 'UTC'),
                deleted_at   TIMESTAMP(0)
            );
        ");

        DB::statement("…
        ");

        DB::statement("…
        ");
    }

    /** …
    public function down(){
        DB::statement("DROP TABLE holidays_limits");
        DB::statement("DROP TABLE holidays");
        DB::statement("DROP TABLE companies");
    }
}
```

Figura 4.12: *Migrations* para tabla *holidays*

En el segundo fichero, está el código para crear los tipos ENUM correspondientes a *enum_event_types* y a *enum_event_states*, ambos añadidos a la tabla de *holidays* junto con una nueva columna, el nombre del evento.

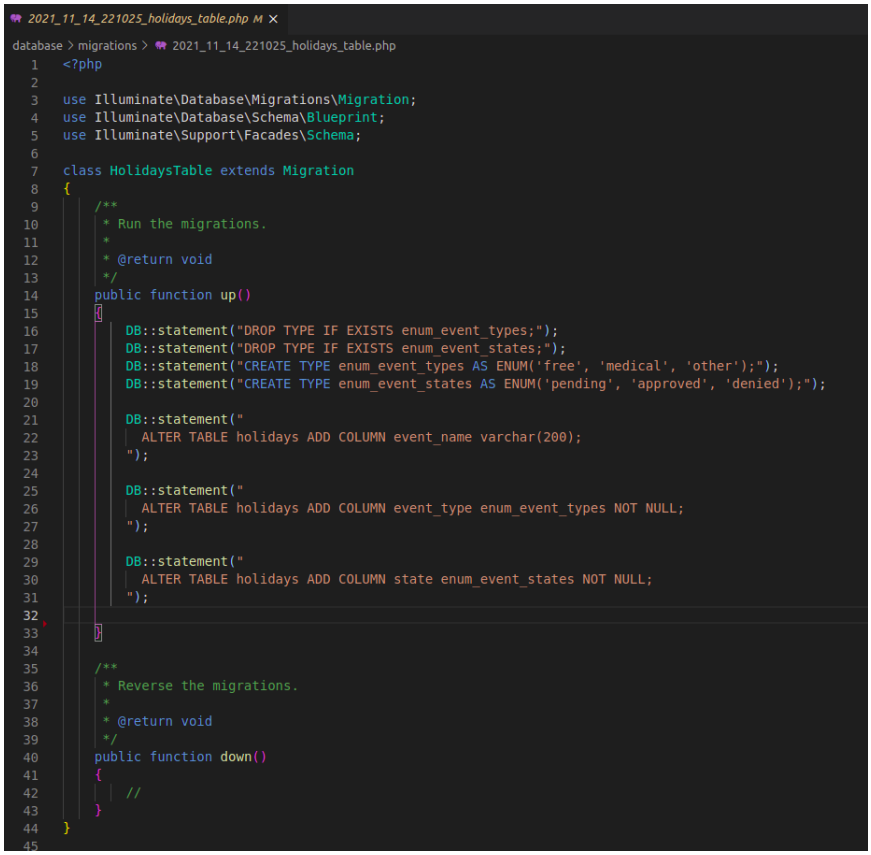


Figura 4.13: *Migrations* para tabla *holidays*

Creación de comando para usuario administrador

Laravel también permite la creación de comandos personalizados [59] para realizar las acciones más primarias. Este es el caso de la creación de un usuario administrador dentro de este proyecto, dado que no se permite el auto-registro.
La descripción de la clase necesaria para poder ejecutar este comando se muestra en la **Figura 4.14**.

app > Console > Commands > UserCommand.php

```php
use Illuminate\Console\Command;
use App\Models\User;
use Hash;


class UserCommand extends Command
{
    /**…
    protected $signature = 'users:create';

    /**…
    protected $description = 'Command to create admin user';

    /**…
    public function __construct()
    {
        parent::__construct();
    }

    /**
     * Execute the console command.
     *
     * @return int
     */
    public function handle()
    {
        $new_user = new User;

        $new_user->first_name = $this->ask('First name: ');
        $new_user->last_name = $this->ask('Last name: ');
        $new_user->phone = $this->ask('Phone number: ');
        $new_user->email = $this->ask('Email: ');
        $new_user->password = Hash::make($this->secret('Choose a password: '));
        $new_user->admin = true;
        $new_user->event_visibility = true;

        $new_user->save();


        $this->info('User created succesfully');
    }
}

```

Figura 4.14: *Users create command*

En concreto, se crea un usuario, al que se asigna un primer y segundo nombre, según lo introducido por línea de comandos, lo mismo para el teléfono y el email, así como la contraseña. Este usuario será siempre administrador y sus eventos son por defecto visibles.

Para llamar a este comando se ejecuta la orden

```
php artisan users:create
```

que mostrará las preguntas necesarias para asignar los valores correspondientes al usuario.

Capítulo 5

Pruebas

En este capítulo se describen las pruebas realizadas durante el desarrollo del proyecto para asegurar la correspondencia del sistema con lo solicitado por el cliente.
En concreto, el contenido del capítulo se puede dividir en los siguientes apartados:

- Una **introducción teórica** a las pruebas de desarrollo *software*.
- Una descripción de los **tipos de pruebas** que se han tenido en cuenta en este proyecto.
- Las **pruebas realizadas** en el mismo.

5.1. Introducción teórica

Se conocen como pruebas de *software* [60] al proceso o actividades dentro del desarrollo en que se comprueba que el producto desarrollado sea correcto y cumpla con los requisitos establecidos. Sirven además, para garantizar que las funcionalidades se corresponden con las esperadas y se basan en la NORMA ISO 9126 [1], un estándar internacional para evaluar la calidad de un producto, en este caso *software* [61].
Los aspectos que se miden mediante las pruebas de *software* son:

- **Calidad interna**: Que puede medirse tanto inspeccionando el código como revisando la documentación, sin que esto dependa de la ejecución del *software*.
- **Calidad externa**: En este caso se mide a partir del comportamiento del producto, es decir, durante su ejecución.
- **Calidad en uso**: Esta métrica se obtiene cuando el producto es usado por el usuario o cliente.

[1] `https://www.issco.unige.ch/en/research/projects/ewg96/node14.html#SECTION00311000000000000000`

En concreto, la NORMA ISO 9126 establece que las características que determinan la calidad del *software* son: la funcionalidad, la fiablidad, la usabilidad, la eficiencia, la mantenibilidad y la portabilidad. De estas las que más nos interesa evaluar en este proyecto son:

- La **funcionalidad**: Que se relaciona con la existencia de un conjunto de funciones y propiedades específicas que satisfacen las necesidades implícitas o explícitas.
- La **usabilidad**: Que se relaciona con el esfuerzo necesario para su uso por un conjunto de usuarios determinado.

Además de los tipos en que se van a dividir las pruebas en este capítulo, puede distinguirse entre pruebas manuales y pruebas automatizadas [62]:

- Las **pruebas manuales** son llevadas a cabo por una persona o equipo que se encarga de comprobar que el sistema se comporte según lo previsto.
- Las **pruebas automatizadas** pueden ir desde comprobar que una sección de código aislado se comporte de forma correcta, hasta simular la funcionalidad completa del sistema como si fuera un humano quien lo estuviese usando.

Realizar pruebas antes de poner en uso un producto *software* reduce los costes de mantenimiento del mismo, además de garantizar una mayor estabilidad a la hora de desarrollar funciones nuevas y permite identificar defectos y ayudar a la toma de decisiones futuras.
Algunas técnicas utilizadas para describir pruebas son [63]:

- **Particionamiento de equivalencia**: Los valores de entrada se dividen en subconjuntos de los que se espera que presenten un comportamiento similar, de forma que solo sea necesario probar un valor de cada subconjunto. Estas particiones pueden aplicarse a datos válidos o que deben aceptarse (pruebas positivas) o a valores que deben rechazarse (pruebas negativas).
- **Análisis de los valores límite**: Consiste en probar los valores extremos de las particiones. También pueden existir límites válidos y límites no válidos.
- **Transición de estados**: Este tipo de pruebas se utiliza cuando en ciertos aspectos el sistema podría representarse como una "máquina de estados finita".

Además de esto, hay que tener en cuenta los diferentes niveles según el punto de vista desde el que se examinen las funciones *software*: pruebas unitarias, de integración y de sistema o *End to End*; así como la importancia que se le debe dar a cada una de ellas.
Para establecer esto último, las metodologías ágiles parten de la conocida como pirámide de Cohn o pirámide de automatización de pruebas ágiles [64] [65] [66], que el propio Mike Cohn propone en su libro *Succeeding with Agile: Software Development Using Scrum*. Ésta sitúa en la base las pruebas unitarias o *unit tests*, seguidas por las pruebas de integración y después las pruebas End to End, dando menor importancia a las pruebas de Interfaz de Usuario (GUI).

Esto no implica que la interfaz o la experiencia de usuario sean menos importantes, sino que con las pruebas lo que se pretende es prevenir los errores en lugar de encontrarlos. Dicho de otra forma, los *bugs* que se encuentran en la interfaz de usuario o en el diseño no tienen por qué afectar a la lógica de la aplicación, sin embargo, los errores que se pueden corregir realizando pruebas unitarias o de integración podrían suponer errores a niveles más altos y desencadenar consecuencias mucho mayores, tanto en la lógica como en el diseño.

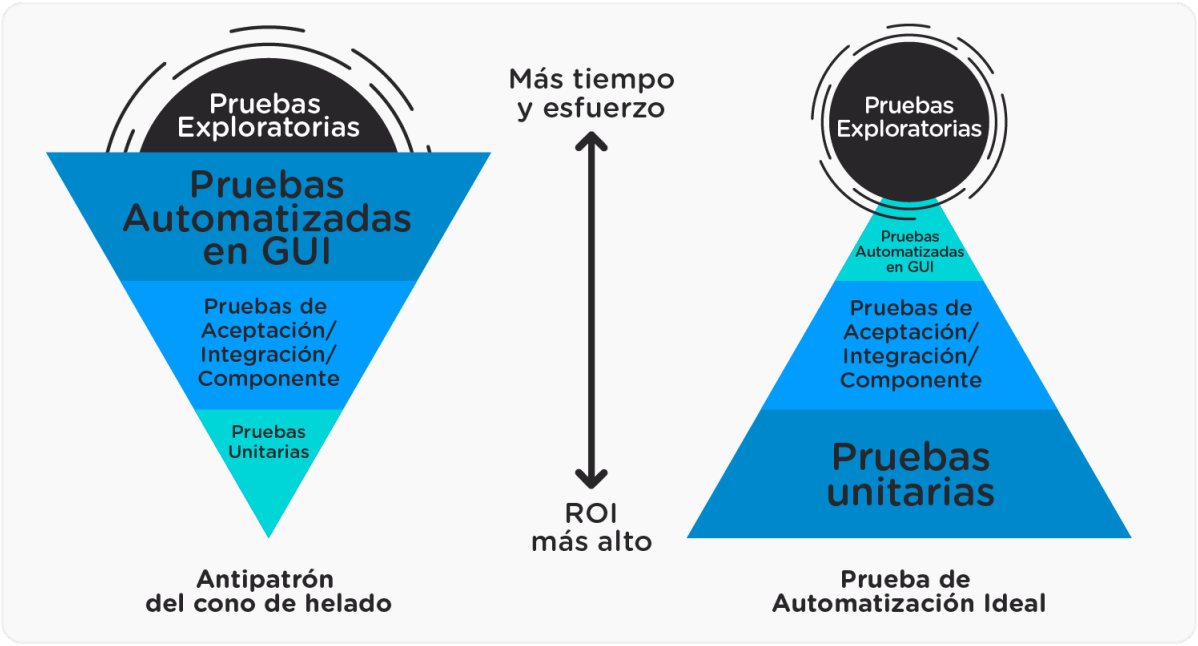


Figura 5.1: Pirámide de Cohn

5.1.1. Tipos de pruebas

Pruebas unitarias

Las pruebas unitarias [62] [67], que según la pirámide deben realizarse en mayor cantidad, examinan el *software* desde un nivel básico o fundamental. Consisten en comprobar la corrección del *software* a la hora de ejecutar unidades individuales de código, normalmente funciones o métodos de código independientes. Para ello se ejecutan las funciones en un entorno de prueba con una entrada simulada y se compara la salida con la esperada dada dicha entrada. Si la salida se corresponde con la esperada, entonces la prueba tiene éxito o se supera. En caso contrario, no se supera.

Pruebas de integración

A diferencia de las anteriores, estas pruebas [62] [67] se encargan de comprobar la corrección de más de una unidad de *software* cuando estas están relacionadas entre sí, con el objetivo de asegurar que el conjunto presenta el comportamiento deseado. Es decir, buscan identificar fallos cuando dos o más elementos del *software* se combinan o comunican entre sí y asegurar que las interacciones entre estos cumplen las especificaciones. Estas deben ejecutarse una vez que se han realizado todas las pruebas unitarias y se han finalizado con éxito.

Pruebas *End to End*

Estas [62] [67] son las últimas a realizar y las menos ejecutadas, ya que son costosas, difíciles de preparar y mantener y las que requieren más tiempo. Este tipo de pruebas tiene como objetivo probar el sistema como un conjunto y verificar que todo funcione correctamente. Se basan en los requisitos generales o en las historias de usuario y abarca todas las partes combinadas del sistema.

- **Pruebas de caja negra:** En este caso, no se considera el código dentro de los aspectos a evaluar, no están basadas en el conocimiento del diseño interno del programa, sino que se enfocan en los requisitos y funcionalidad del sistema. Se enfocan las pruebas de forma que el *software* es una caja negra con entradas y salidas, sin conocimiento de la estructura interna del programa. Es decir, importa lo que hace el sistema, no cómo.
- **Pruebas de caja blanca:** A diferencia de las pruebas de caja negra, las pruebas de caja blanca se basan en los distintos escenarios que se puedan generar por las estructuras condicionales, estados, etc. Es decir, en el conocimiento del diseño interno o el código del sistema.

Pruebas de usabilidad

Además de las anteriores, se pueden realizar pruebas de usabilidad. Estas están centradas en el usuario [68] y tienen como objetivo evaluar el producto final realizando una serie de pruebas con usuarios reales o con un perfil lo más similar posible al de usuario final. Consisten en seleccionar a un grupo de usuarios y solicitarles que lleven a cabo una serie de tareas para las que fue diseñada la aplicación. Entre las métricas que se pueden medir durante estas pruebas se encuentra el tiempo, la respuesta emocional del usuario, el número de errores que cometan, etc.

5.2. Pruebas realizadas

Para describir las pruebas se utilizarán los siguientes campos:

- **Identificador**: Número único asociado al caso de prueba correspondiente.
- **Descripción**: Explicación del objetivo y el contexto referentes al caso de prueba.

- **Precondiciones**: Contexto en el que debe desarrollarse la prueba, datos necesarios, entrada utilizada, etc.
- **Criticidad**: Relevancia del caso de prueba para el sistema: Alta, media o baja.
- **Acción realizada**: Paso o pasos a ejecutar para realizar la prueba correspondiente.
- **Resultado esperado**: Valores esperados o cambio de estado deseado para que el resultado de la prueba sea satisfactorio.
- **Resultado obtenido**: Valores o cambio de estado resultantes de ejecutar los pasos correspondientes a la prueba.

Identificador	CB - 0001
Descripción	Inicio de una nueva sesión introduciendo los datos de un usuario registrado: par (email, contraseña) correctos.
Precondiciones	El usuario utilizado para iniciar sesión ya está registrado en el sistema y la sesión no ha sido iniciada previamente.
Criticidad	Alta.
Acción realizada	El usuario introduce el valor del email correcto en el campo del formulario correspondiente y la contraseña correcta en el campo correspondiente, después acciona el botón de “Iniciar sesión”.
Resultado esperado	Se muestra la vista de “Summary” correspondiente al usuario registrado.
Resultado obtenido	Se muestra la vista de “Summary” correspondiente al usuario registrado.

Identificador	CB - 0002
Descripción	Inicio de una nueva sesión introduciendo los datos de un usuario registrado: par (email, contraseña) incorrectos.
Precondiciones	El usuario utilizado para iniciar sesión ya está registrado en el sistema y la sesión no ha sido iniciada previamente.
Criticidad	Alta.
Acción realizada	El usuario introduce el valor del email correcto en el campo del formulario correspondiente y la contraseña incorrecta en el campo correspondiente, después acciona el botón de “Iniciar sesión”.
Resultado esperado	Se muestra un mensaje de error y el usuario permanece en la vista de “Login”.
Resultado obtenido	Se muestra el mensaje de error: “These credentials do not match our records” y el usuario permanece en la vista de “Login”.

Identificador	CB - 0003
Descripción	Inicio de una nueva sesión introduciendo los datos de un usuario no registrado.
Precondiciones	El usuario utilizado para iniciar sesión no está registrado en el sistema y la sesión no ha sido iniciada previamente.
Criticidad	Alta.
Acción realizada	El usuario introduce el valor del email incorrecto en el campo del formulario correspondiente y la contraseña en el campo correspondiente, después acciona el botón de “Iniciar sesión”.
Resultado esperado	Se muestra un mensaje de error y el usuario permanece en la vista de “Login”.
Resultado obtenido	Se muestra el mensaje de error: “These credentials do not match our records” y el usuario permanece en la vista de “Login”.

Identificador	CB - 0004
Descripción	Inicio de sesión introduciendo los datos de un usuario administrador correctos.
Precondiciones	El usuario utilizado para iniciar sesión está registrado en el sistema y es administrador.
Criticidad	Alta.
Acción realizada	El usuario introduce el valor del email correcto en el campo del formulario correspondiente y la contraseña correcta en el campo correspondiente, después acciona el botón de “Iniciar sesión”.
Resultado esperado	Se muestra la vista de “Summary” correspondiente al usuario que ha iniciado sesión y en la barra lateral se muestran las etiquetas de “Summary”, “Timer”, “Calendar” (“My Calendar” y “Company Calendar”), “Reports”, “Management” y “My Profile”.
Resultado obtenido	Se muestra la vista de “Summary” del usuario administrador que ha iniciado sesión y puede ver todas las secciones que le corresponden.

Identificador	CB - 0005
Descripción	Inicio de sesión introduciendo los datos de un usuario no administrador correctos.
Precondiciones	El usuario utilizado para iniciar sesión está registrado en el sistema y no es administrador.
Criticidad	Alta.
Acción realizada	El usuario introduce el valor del email correcto en el campo del formulario correspondiente y la contraseña correcta en el campo correspondiente, después acciona el botón de "Iniciar sesión".
Resultado esperado	Se muestra la vista de "Summary" correspondiente al usuario que ha iniciado sesión y en la barra lateral se muestran las etiquetas de "Summary", "Timer", "Calendar" ("My Calendar" y "Company Calendar") y "My Profile". Si intenta acceder a las rutas de "Reports" o "Management" se le deniega el acceso.
Resultado obtenido	Se muestra la vista de "Summary" correspondiente al usuario que ha iniciado sesión y en la barra lateral se muestran las etiquetas de "Summary", "Timer", "Calendar" ("My Calendar" y "Company Calendar") y "My Profile". Si intenta acceder a las rutas de "Reports" o "Management" se le deniega el acceso.

Identificador	CB - 0006
Descripción	El usuario registrado accede a la vista de "Summary" o de "Timer" y revisa las horas trabajadas, las horas planificadas y las horas extra en la barra superior.
Precondiciones	El usuario ha iniciado sesión previamente.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de "Summary" o a la vista de "Timer" y selecciona las horas en la barra superior para el día, la semana o el mes.
Resultado esperado	Las horas trabajadas son correctas y si son superiores a las planificadas esta diferencia se representa correctamente en las horas extra, tanto para las horas de hoy, como para las de la semana o el mes.
Resultado obtenido	Las horas trabajadas son correctas y si son superiores a las planificadas esta diferencia se representa correctamente en las horas extra, tanto para las horas de hoy, como para las de la semana o el mes.

Identificador	CB - 0007
Descripción	El usuario registrado accede a la vista de “Summary”y revisa el porcentaje de horas trabajados durante el día o la semana respecto a las planificadas.
Precondiciones	El usuario ha iniciado sesión previamente.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Summary”.
Resultado esperado	El porcentaje mostrado en los gráficos de las horas trabajadas respecto a las planificadas se corresponde con la realidad.
Resultado obtenido	El porcentaje mostrado en los gráficos de las horas trabajadas respecto a las planificadas se corresponde con la realidad.

Identificador	CB - 0008
Descripción	El usuario registrado accede a la vista de “Summary”y revisa la tabla de eventos.
Precondiciones	El usuario ha iniciado sesión previamente.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Summary”.
Resultado esperado	La tabla de eventos muestra un resumen de los eventos del usuario correspondientes. En caso de no haber ninguno se muestra un mensaje indicándolo.
Resultado obtenido	La tabla de eventos muestra un resumen de los eventos del usuario correspondientes. En caso de no haber ninguno se muestra un mensaje indicándolo.

Identificador	CB - 009
Descripción	El usuario registrado accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior con la intención de iniciar un registro de “Trabajo”.
Precondiciones	El usuario ha iniciado sesión previamente y el último registro es de “Trabajo”.
Criticidad	Alta.
Acción realizada	El usuario accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior, que en este caso mostrará el icono de pausa, al tiempo que el contador tendrá un reborde de color verde oscuro.
Resultado esperado	Se inicia un registro de “Descanso”, ya que el botón accionado es el de pausa y el reborde del contador, que vuelve a contar desde cero, pasa a ser naranja.
Resultado obtenido	Se inicia un registro de “Descanso”, ya que el botón accionado es el de pausa y el reborde del contador, que vuelve a contar desde cero, pasa a ser naranja.

Identificador	CB - 0010
Descripción	El usuario registrado accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior con la intención de iniciar un registro de “Trabajo”.
Precondiciones	El usuario ha iniciado sesión previamente y el último registro es de “Descanso”.
Criticidad	Alta.
Acción realizada	El usuario accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior, que en este caso mostrará el icono de play, al tiempo que el contador tendrá un reborde de color naranja.
Resultado esperado	Se inicia un registro de “Trabajo” y el reborde del contador, que vuelve a contar desde cero, pasa a ser verde oscuro.
Resultado obtenido	Se inicia un registro de “Trabajo” y el reborde del contador, que vuelve a contar desde cero, pasa a ser verde oscuro.

Identificador	CB - 0011
Descripción	El usuario registrado accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior con la intención de iniciar un registro de “Trabajo”.
Precondiciones	El usuario ha iniciado sesión previamente y el último registro es de “Fuera de la oficina”.
Criticidad	Alta.
Acción realizada	El usuario accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior, que en este caso mostrará el icono de play, al tiempo que el contador tendrá un reborde de color amarillo.
Resultado esperado	Se inicia un registro de “Trabajo” y el reborde del contador, que vuelve a contar desde cero, pasa a ser verde oscuro.
Resultado obtenido	Se inicia un registro de “Trabajo” y el reborde del contador, que vuelve a contar desde cero, pasa a ser verde oscuro.

Identificador	CB - 0012
Descripción	El usuario registrado accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior con la intención de iniciar un registro de “Descanso”.
Precondiciones	El usuario ha iniciado sesión previamente y el último registro es de “Fuera de la oficina”.
Criticidad	Alta.
Acción realizada	El usuario accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior, que en este caso mostrará el icono de play, al tiempo que el contador tendrá un reborde de color amarillo.
Resultado esperado	Se crea un registro de “Trabajo” y el reborde del contador pasa de color amarillo a verde oscuro, ya que el registro anterior es de “Fuera de la oficina” y la lógica de la aplicación impide realizar descansos fuera de las horas de trabajo.
Resultado obtenido	Se crea un registro de “Trabajo” y el reborde del contador pasa de color amarillo a verde oscuro, ya que el registro anterior es de “Fuera de la oficina” y la lógica de la aplicación impide realizar descansos fuera de las horas de trabajo.

Identificador	CB - 0013
Descripción	El usuario registrado accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior con la intención de iniciar un registro de “Descanso”.
Precondiciones	El usuario ha iniciado sesión previamente y el último registro es de “Trabajo”.
Criticidad	Alta.
Acción realizada	El usuario accede a la vista de “Summary” o de “Timer” y acciona el botón principal de la barra superior, que en este caso mostrará el icono de pause, al tiempo que el contador tendrá un reborde de color verde oscuro.
Resultado esperado	Se crea un registro de “Descanso” y el reborde del contador pasa de color verde oscuro a naranja.
Resultado obtenido	Se crea un registro de “Descanso” y el reborde del contador pasa de color verde oscuro a naranja.

Identificador	CB - 0014
Descripción	El usuario registrado accede a la vista de “Summary” o de “Timer” y acciona el botón secundario superior de la barra superior con la intención de iniciar un registro de “Fuera de la oficina”.
Precondiciones	El usuario ha iniciado sesión previamente y el último registro es de “Descanso”.
Criticidad	Alta.
Acción realizada	El usuario accede a la vista de “Summary” o de “Timer” y acciona el botón secundario superior de la barra superior, que mostrará el icono de stop, al tiempo que el contador tendrá un reborde de color naranja.
Resultado esperado	No se permite detener el registro y se muestra un mensaje de error.
Resultado obtenido	No se permite detener el registro y se muestra un mensaje de error: ‘Can’t stop a pause record’.

Identificador	CB - 0015
Descripción	El usuario registrado accede a la vista de “Summary” o de “Timer” y acciona el botón secundario superior de la barra superior con la intención de iniciar un registro de “Fuera de la oficina”.
Precondiciones	El usuario ha iniciado sesión previamente y el último registro es de “Trabajo”.
Criticidad	Alta.
Acción realizada	El usuario accede a la vista de “Summary” o de “Timer” y acciona el botón secundario superior de la barra superior, que mostrará el icono de stop, al tiempo que el contador tendrá un reborde de color verde oscuro.
Resultado esperado	Se inicia un registro de “Fuera de la oficina” y el contador se detiene (mostrando el valor 00:00:00) y pasa de tener un reborde color verde oscuro a tener un reborde amarillo.
Resultado obtenido	Se inicia un registro de “Fuera de la oficina” y el contador se detiene (mostrando el valor 00:00:00) y pasa de tener un reborde color verde oscuro a tener un reborde amarillo.

Identificador	CB - 0016
Descripción	El usuario registrado accede a la vista de “Summary” desdse la vista de “Timer” o viceversa.
Precondiciones	El usuario ha iniciado sesión previamente y el contador está mostrando el valor de un registro de tipo “Trabajo” o “Descanso”.
Criticidad	Media Alta.
Acción realizada	El usuario accede a la vista de “Summary” desde la vista de “Timer” o viceversa, con un registro de tipo “Trabajo” o “Descanso” en curso.
Resultado esperado	El valor del contador mostrado en la vista de “Summary” es igual o superior en un segundo al mostrado en la vista de “Timer”. Lo mismo ocurre si se pasa de la vista de “Summary” a la de “Timer”.
Resultado obtenido	El valor del contador mostrado en la vista de “Summary” es igual o superior en uno o dos segundos al mostrado en la vista de “Timer”. Lo mismo ocurre si se pasa de la vista de “Summary” a la de “Timer”.

Identificador	CB - 0017
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Lista y genera un registro de “Descanso” o uno de “Trabajo” después de uno de “Descanso”.
Precondiciones	El usuario ha iniciado sesión previamente y el registro que se está ejecutando es de “Descanso” o de “Trabajo”.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Lista. Acciona el botón principal del menú superior con el objetivo de iniciar un registro de “Trabajo” si el anterior era de “Descanso” o uno de “Descanso” si el anterior era de “Trabajo”.
Resultado esperado	Se genera un nuevo elemento en la vista en tiempo real que muestra el registro anterior al iniciado con su tiempo de duración y las horas de inicio y finalización del mismo, además de mostrar un color diferente según sea un registro de “Trabajo” (verde) o de “Descanso” (naranja).
Resultado obtenido	Se genera un nuevo elemento en la vista en tiempo real que muestra el registro anterior al iniciado con su tiempo de duración y las horas de inicio y finalización del mismo, además de mostrar un color diferente según sea un registro de “Trabajo” (verde) o de “Descanso” (naranja).

Identificador	CB - 0018
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Lista y selecciona la modificación de un registro de forma que esto lo haga incompatible con otros registros.
Precondiciones	El usuario ha iniciado sesión previamente y ha generado registros.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Lista. Selecciona la modificación de un registro de tipo “Descanso” comprendido entre dos registros de tipo “Trabajo” de forma que quede espacio entre este registro y uno de los anteriores.
Resultado esperado	Si la modificación no es anterior al tiempo de inicio ni posterior al tiempo de finalización del registro a modificar, se permite.
Resultado obtenido	Si la modificación no es anterior al tiempo de inicio ni posterior al tiempo de finalización del registro a modificar, se permite.

Identificador	CB - 0019
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Lista y selecciona la modificación de un registro de forma correcta.
Precondiciones	El usuario ha iniciado sesión previamente y ha generado registros.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Lista. Selecciona la modificación de un registro de tipo “Trabajo” de forma que no lo haga incompatible con registros anteriores o posteriores.
Resultado esperado	Se efectúa la modificación del registro, que se actualiza en el elemento correspondiente de la vista.
Resultado obtenido	Se efectúa la modificación del registro, que se actualiza en el elemento correspondiente de la vista.

Identificador	CB - 0020
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Lista y selecciona la eliminación de un registro de forma incorrecta.
Precondiciones	El usuario ha iniciado sesión previamente y ha generado registros.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Lista. Selecciona la eliminación de un registro de tipo “Trabajo” que afecte a uno de tipo “Descanso”.
Resultado esperado	Se genera un error indicando que la eliminación del registro no es posible.
Resultado obtenido	Se muestra el mensaje de error: “You can only delete lonely working records!”.

Identificador	CB - 0021
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Lista y selecciona la eliminación de un registro de forma correcta.
Precondiciones	El usuario ha iniciado sesión previamente y ha generado registros.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Lista. Selecciona la eliminación de un registro de tipo “Trabajo” que no esté rodeado por otros de tipo “Descanso”.
Resultado esperado	El elemento correspondiente al registro desaparece de la vista.
Resultado obtenido	El elemento correspondiente al registro desaparece de la vista.

Identificador	CB - 0022
Descripción	El usuario registrado accede a la vista de “Timer” y selecciona la vista “Daily” en el desplegable.
Precondiciones	El usuario ha iniciado sesión previamente y ha generado registros en la fecha actual.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Daily en el desplegable.
Resultado esperado	Los registros de la fecha actual aparecen en forma de horario, mostrando los de tipo “Trabajo” en verde y los de tipo “Descanso” en naranja.
Resultado obtenido	Los registros de la fecha actual aparecen en forma de horario, mostrando los de tipo “Trabajo” en verde y los de tipo “Descanso” en naranja.

Identificador	CB - 0023
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Daily y genera un registro accionando los botones del menú superior.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Daily. Acciona los botones del menú superior generando un registro.
Resultado esperado	El elemento correspondiente al registro aparece en la vista en su rango horario correspondiente, con color verde si es un registro de tipo “Trabajo” o naranja si es de tipo “Descanso”.
Resultado obtenido	El elemento correspondiente al registro aparece en la vista en su rango horario correspondiente, con color verde si es un registro de tipo “Trabajo” o naranja si es de tipo “Descanso”.

Identificador	CB - 0024
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Daily y accede a los registros de fechas anteriores.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Daily. Acciona las flechas para desplazarse de unas fechas a otras que sean anteriores al día actual.
Resultado esperado	Se muestran los registros correspondientes a las fechas seleccionadas.
Resultado obtenido	Se muestran los registros correspondientes a las fechas seleccionadas.

Identificador	CB - 0025
Descripción	El usuario registrado accede a la vista de “Timer” en la vista de Daily e intenta acceder a los registros de fechas posteriores.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Daily. Acciona las flechas para desplazarse a fechas posteriores a la actual.
Resultado esperado	La flecha aparece en color gris y pulsarla no tiene ningún efecto.
Resultado obtenido	La flecha aparece en color gris y pulsarla no tiene ningún efecto.

Identificador	CB - 0026
Descripción	El usuario registrado accede a la vista de “Timer” y selecciona la vista Weekly en el desplegable.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Weekly en el desplegable.
Resultado esperado	Se muestran los registros de la semana, a partir del primer día de la semana hasta la fecha actual.
Resultado obtenido	Se muestran los registros de la semana, a partir del primer día de la semana hasta la fecha actual.

Identificador	CB - 0027
Descripción	El usuario registrado accede a la vista de “Timer”, selecciona la vista Weekly en el desplegable y genera un registro.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Weekly en el desplegable. A continuación acciona los botones del menú superior generando un registro.
Resultado esperado	La vista de la semana se actualiza en el día correspondiente mostrando los registros de dicha fecha, incluido el que se acaba de generar.
Resultado obtenido	La vista de la semana se actualiza en el día correspondiente mostrando los registros de dicha fecha, incluido el que se acaba de generar.

Identificador	CB - 0028
Descripción	El usuario registrado accede a la vista de “Timer”, selecciona la vista Weekly en el desplegable y se desplaza a las semanas anteriores.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Weekly en el desplegable. A continuación acciona las flechas para acceder a los registros de semanas anteriores.
Resultado esperado	Se muestran los registros de semanas anteriores.
Resultado obtenido	Se muestran los registros de semanas anteriores.

Identificador	CB - 0029
Descripción	El usuario registrado accede a la vista de “Timer”, selecciona la vista Weekly en el desplegable e intenta desplazarse a las semanas posteriores.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Timer” y selecciona la vista de tipo Weekly en el desplegable. A continuación acciona las flechas para acceder a los registros de semanas posteriores.
Resultado esperado	La flecha se muestra en color gris y pulsarla no tiene ningún efecto.
Resultado obtenido	La flecha se muestra en color gris y pulsarla no tiene ningún efecto.

Identificador	CB - 0030
Descripción	El usuario no administrador accede a la vista de “My Calendar” y visualiza el resumen de sus eventos en la barra superior.
Precondiciones	El usuario ha iniciado sesión y no es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “My Calendar”.
Resultado esperado	Los eventos correctos se visualizan en la barra superior.
Resultado obtenido	Los eventos correctos se visualizan en la barra superior.

Identificador	CB - 0031
Descripción	El usuario no administrador accede a la vista de “My Calendar” y visualiza el calendario con sus eventos.
Precondiciones	El usuario ha iniciado sesión y no es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “My Calendar” y se desplaza en el calendario para ver los diferentes meses. Pulsa sobre alguna de las fechas en las que haya un evento.
Resultado esperado	Se muestran las fechas en las que haya algún evento con el color correspondiente y al pulsar la fecha se muestra el nombre del evento.
Resultado obtenido	Se muestran las fechas en las que haya algún evento con el color correspondiente y al pulsar la fecha se muestra el nombre del evento.

Identificador	CB - 0032
Descripción	El usuario accede a la vista de “My Calendar” y solicita añadir un nuevo evento en una fecha incorrecta.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “My Calendar” y pulsa en el botón para añadir un nuevo evento. Selecciona una fecha anterior a la actual.
Resultado esperado	Se muestra un mensaje indicando que no se puede agregar un nuevo evento en la fecha seleccionada.
Resultado obtenido	Se muestra el mensaje de error: ‘Dates must be after actual date!’

Identificador	CB - 0033
Descripción	El usuario accede a la vista de “My Calendar” y solicita añadir un nuevo evento en una fecha incorrecta.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “My Calendar” y pulsa en el botón para añadir un nuevo evento. Selecciona una fecha en la que otro usuario con funciones similares ya ha solicitado un evento.
Resultado esperado	Se muestra un mensaje indicando que no se puede agregar un nuevo evento en la fecha seleccionada por incompatiblidades con otro usuario.
Resultado obtenido	Se muestra el mensaje de error: ‘A similar user has an event near to these dates!’

Identificador	CB - 0034
Descripción	El usuario accede a la vista de “My Calendar” y solicita añadir un nuevo evento en una fecha correcta.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “My Calendar” y pulsa en el botón para añadir un nuevo evento. Selecciona una fecha en la que no hay eventos de otro usuario similar y posterior a la fecha actual.
Resultado esperado	El evento se agrega al calendario del usuario correspondiente.
Resultado obtenido	El evento se agrega al calendario del usuario correspondiente.

Identificador	CB - 0035
Descripción	El usuario administrador accede a la vista de “My Calendar” y visualiza el calendario con sus eventos y los de otros usuarios.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “My Calendar” y se desplaza en el calendario para ver los diferentes meses. Pulsa sobre alguna de las fechas en las que haya un evento.
Resultado esperado	Se muestran las fechas en las que haya algún evento con el color correspondiente si son eventos del usuario loggeado y al pulsar la fecha se muestra el nombre del evento y el usuario al que corresponde en caso de ser eventos de otros usuarios.
Resultado obtenido	Se muestran las fechas en las que haya algún evento con el color correspondiente si son eventos del usuario loggeado y al pulsar la fecha se muestra el nombre del evento y el usuario al que corresponde en caso de ser eventos de otros usuarios.

Identificador	CB - 0036
Descripción	El usuario no administrador accede a la vista de “Company Calendar” y visualiza el calendario con los eventos de la compañía.
Precondiciones	El usuario ha iniciado sesión y no es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Company Calendar” y visualiza los eventos en la tabla de la compañía.
Resultado esperado	Se muestran solo los eventos de la semana correspondientes a esos usuarios que hayan seleccionado que sus eventos sean visibles para el resto.
Resultado obtenido	Se muestran solo los eventos de la semana correspondientes a esos usuarios que hayan seleccionado que sus eventos sean visibles para el resto.

Identificador	CB - 0037
Descripción	El usuario no administrador accede a la vista de “Company Calendar” y visualiza el calendario con los eventos de los usuarios con funciones similares.
Precondiciones	El usuario ha iniciado sesión y no es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Company Calendar” y pulsa el botón para mostrar solo los eventos de usuarios similares.
Resultado esperado	Se muestran solo los eventos de la semana correspondientes a esos usuarios que hayan seleccionado que sus eventos sean visibles para el resto y que tengan funciones similares a los del usuario loggeado.
Resultado obtenido	Se muestran solo los eventos de la semana correspondientes a esos usuarios que hayan seleccionado que sus eventos sean visibles para el resto y que tengan funciones similares a los del usuario loggeado.

Identificador	CB - 0038
Descripción	El usuario administrador accede a la vista de “Reports” y visualiza la tabla de registros.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Reports”.
Resultado esperado	Se muestran los registros de todos los usuarios del sistema que no sean administradores correctamente.
Resultado obtenido	Se muestran los registros de todos los usuarios del sistema que no sean administradores correctamente.

Identificador	CB - 0039
Descripción	El usuario administrador accede a la vista de “Reports” y filtra por valor en la tabla de registros.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Reports” y escribe un valor en el filtro.
Resultado esperado	Se muestran los registros de todos los usuarios del sistema donde una de las dos primeras columnas contenga el valor indicado.
Resultado obtenido	Se muestran los registros de todos los usuarios del sistema donde una de las dos primeras columnas contenga el valor indicado.

Identificador	CB - 0040
Descripción	El usuario administrador accede a la vista de “Reports” y solicita exportar los datos a formato .csv.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Reports”, selecciona el formato .csv y pulsa el botón para exportar la tabla.
Resultado esperado	Se genera un archivo .csv con los datos de los registros de todos los usuarios del sistema no administradores.
Resultado obtenido	Se genera un archivo .csv con los datos de los registros de todos los usuarios del sistema no administradores.

Identificador	CB - 0041
Descripción	El usuario administrador accede a la vista de “Reports” y solicita exportar los datos a formato .pdf.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Reports”, selecciona el formato .pdf y pulsa el botón para exportar la tabla.
Resultado esperado	Se genera un archivo .pdf con los datos de los registros de todos los usuarios del sistema no administradores.
Resultado obtenido	Se genera un archivo .pdf con los datos de los registros de todos los usuarios del sistema no administradores.

Identificador	CB - 0042
Descripción	El usuario administrador accede a la vista de “Management” y visualiza la tabla de usuarios.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Management”.
Resultado esperado	Se muestran todos los usuarios del sistema junto con sus datos.
Resultado obtenido	Se muestran todos los usuarios del sistema junto con sus datos.

Identificador	CB - 0043
Descripción	El usuario administrador accede a la vista de “Management” y filtra la tabla de usuarios por valor.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Media.
Acción realizada	El usuario accede a la vista de “Management” y escribe un valor en el filtro.
Resultado esperado	Se muestran todos los usuarios del sistema cuyo nombre o identificador contenga el valor indicado y los datos de los mismos.
Resultado obtenido	Se muestran todos los usuarios del sistema cuyo nombre o identificador contenga el valor indicado y los datos de los mismos.

Identificador	CB - 0044
Descripción	El usuario administrador accede a la vista de “Management” y solicita modificar un usuario.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Management” y pulsa en el icono de modificar al lado del registro de un usuario.
Resultado esperado	Se muestra una vista similar a la de Add New User con los datos del usuario en cuestión. El procedimiento a seguir es el mismo que el de añadir un usuario.
Resultado obtenido	Se muestra una vista similar a la de Add New User con los datos del usuario en cuestión. El procedimiento a seguir es el mismo que el de añadir un usuario. El usuario se agrega correctamente

Identificador	CB - 0045
Descripción	El usuario administrador accede a la vista de “Management” y solicita eliminar un usuario.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Management” y pulsa en el icono de eliminar al lado del registro de un usuario.
Resultado esperado	Se eliminan el usuario y los datos correspondientes del sistema.
Resultado obtenido	Se eliminan el usuario y los datos correspondientes del sistema.

Identificador	CB - 0046
Descripción	El usuario administrador accede a la vista de “Management” y solicita añadir un usuario de forma incorrecta.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Management”, pulsa en el botón de añadir usuario y en la vista de “Add New User” introduce un email que no se corresponde con el formato ‘string@string.string’.
Resultado esperado	Se muestra un mensaje de error indicando que el email debe tener un formato concreto.
Resultado obtenido	Se muestra un mensaje de error indicando que el email debe tener un formato concreto. Se muestra el error: ‘Incluye un signo ‘@’ en la dirección de correo electrónico. La dirección xxx no incluye el signo ‘@’ ’ o el error: ‘The email is not valid’ si contiene un @ pero no tiene el formato indicado.

Identificador	CB - 0047
Descripción	El usuario administrador accede a la vista de “Management” y solicita añadir un usuario de forma incorrecta.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Management”, pulsa en el botón de añadir usuario y en la vista de “Add New User” introduce un email que ya está en el sistema.
Resultado esperado	Se muestra un mensaje de error indicando que el email ya está registrado.
Resultado obtenido	Se muestra el mensaje de error: ‘The email is already in database’.

Identificador	CB - 0048
Descripción	El usuario administrador accede a la vista de “Management” y solicita añadir un usuario de forma incorrecta.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Management”, pulsa en el botón de añadir usuario y en la vista de “Add New User” introduce una contraseña diferente en el campo “Password” y en el campo “Pasword confirmation”.
Resultado esperado	Se muestra un mensaje de error indicando que la contraseña y la comprobación de la contraseña deben ser iguales.
Resultado obtenido	Se muestra el mensaje de error: ‘The password confirmation does not match password’.

Identificador	CB - 0049
Descripción	El usuario administrador accede a la vista de “Management” y solicita añadir un usuario de forma correcta.
Precondiciones	El usuario ha iniciado sesión y es administrador.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “Management”, pulsa en el botón de añadir usuario y en la vista de “Add New User” introduce los datos del usuario de forma correcta.
Resultado esperado	Se añade el usuario al sistema y si se ha añadido una imagen de usuario esta se guarda en el directorio correspondiente. Se redirige al usuario administrador a la vista de “Management” donde aparece el nuevo usuario.
Resultado obtenido	Se añade el usuario al sistema y si se ha añadido una imagen de usuario esta se guarda en el directorio correspondiente. Se redirige al usuario administrador a la vista de “Management” donde aparece el nuevo usuario.

Identificador	CB - 0050
Descripción	El usuario accede a la vista de “My Profile” y visualiza sus datos.
Precondiciones	El usuario ha iniciado sesión.
Criticidad	Baja.
Acción realizada	El usuario accede a la vista de “My Profile”.
Resultado esperado	Se muestran los datos y la imagen correspondientes al usuario loggeado.
Resultado obtenido	Se muestran los datos y la imagen correspondientes al usuario loggeado.

5.2.1. Pruebas automatizadas

Laravel incluye un entorno de pruebas pre-configurado que utilizada el *framework* de testing PHPUnit [2]. Esto se explica más detalladamente en la documentación del *framework* [69] [70] [71].

Para ejecutar este tipo de pruebas se ejecuta el comando [72]

```
vendor/bin/phpunit
```

Y para crear una prueba concreta se ejecuta la orden

```
php artisan make:test "path/nombreTest"
```

Para ilustrar el uso de estas pruebas se mostrará el ejemplo de la **Figura 5.2**.

tests > Feature > Http > Controllers > Auth > LoginControllerTest.php

```php
<?php

namespace Tests\Feature\Http\Controllers\Auth;

use Illuminate\Foundation\Testing\RefreshDatabase;
use Illuminate\Foundation\Testing\WithFaker;
use Tests\TestCase;

use Database\Factories\UserFactory;
use App\Models\User;

class LoginControllerTest extends TestCase
{

    /** @test
     * To test if Login view loads correctly
    */
    public function testLoginView(){
        $response = $this->get(route('login'));
        $response->assertStatus(200);
        $response->assertViewIs('auth.login');
    }

    /** @test
     * To test if login view reloads if not data submitted
    */
    public function testLoginNotValidData(){
        $response = $this->post(route('login'), []);
        $response->assertStatus(302);
        $response->assertSessionHasErrors('email');
    }


    /** @test
     * To test if login view reloads if password is not written
    */
    public function testLoginNotValidPassword2(){
        $response = $this->post(route('login'), [
            'email' => 'admin@prueba.com',
            'password' => ''
        ]);
        $response->assertStatus(302);
        $response->assertSessionHasErrors('password');
    }
```

Figura 5.2: Pruebas automatizadas *Login*

[2] `https://phpunit.de/index.html`

En esta, se puede ver que la clase extiende de *TestCase* y que se usan además las clases *UserFactory* y *User*. Las tres primeras funciones acceden a la URL de la vista de *Login* y comprueban, en el primer caso, que esta carga correctamente y en los dos casos siguientes, que se recarga y no permite el acceso a la aplicación si no se introducen datos o si no se introduce ninguna contraseña.

Otro caso de ejemplo interesante dentro de esta clase es el de la **Figura 5.3**, donde se utiliza la clase *UserFactory* para crear usuarios con los que probar el acceso a la aplicación.

```
60      /** @test
61       * To test if dashboard is loaded if email and password are correct
62       * CB - 0001: Inicio de sesión con datos de usuario registrado (email, contraseña) correctos.
63       */
64      public function testLoginValidData(){
65
66          $user = User::factory()->create();
67          $response = $this->post(route('login'), [
68              'email' => $user->email,
69              'password' => 'password'
70          ]);
71          $response->assertRedirect('/dashboard');
72      }
73
```

Figura 5.3: Pruebas automatizadas *Login*

En este caso, se comprueba que el acceso a la aplicación con los datos correctos redirija a la vista de *Summary*. En concreto, se crea un nuevo usuario, que tendrá por defecto la contraseña *password* y se accede a la aplicación con el email del usuario correspondiente y la contraseña predefinida. Después, se comprueba que la URL en la que se encuentra el usuario recién loggeado sea la de *Summary*.

5.2.2. Pruebas de usabilidad

Las pruebas de usabilidad tienen como objetivo verificar que se cumplen los requisitos de usabilidad explicados en la sección de análisis.

Planificación de las pruebas

Esta comprobación se llevó a cabo mediante una serie de pruebas realizadas con usuarios de la empresa con las que se medirán, entre otras, la facilidad de aprendizaje o la eficiencia, asociadas a realizar una serie de tareas con la aplicación.

Para ello, se proporcionarán a los usuarios una serie de instrucciones, que se encuentran en el **Apéndice A** y cuyo objetivo se resume en la tabla que se presenta a continuación. Se solicitará que expliquen todo aquello que vayan realizando para completar las tareas indicadas y tomando nota tanto de esto, como de sus reacciones. Finalmente, se completará una segunda tabla con los resultados obtenidos, algunos de los cuales se explicarán en más detalle en el **Apéndice A**, donde se señalarán algunas de las mejoras extraídas tras ver las reacciones de los usuarios.

Concretamente, la **eficiencia** se medirá en el tiempo y número de pasos que necesite cada usuario para realizar las tareas, mientras que la **facilidad de aprendizaje** se medirá en el tiempo necesario antes de llevar a cabo la tarea, las dudas que le pueda ocasionar o si directamente no es capaz de realizarla, así como en los pasos erróneos que realice antes de encontrar los correspondientes a la tarea. Estas dos características son las más importantes, ya que ninguno de los usuarios estará familiarizado con la aplicación y todos ellos van a ser usuarios frecuentes de la misma.

Además de estas pruebas, se solicitará a los usuarios que, tras la realización de las pruebas de usablidad, respondan a un cuestionario con 10 preguntas conocido como escala de usablidad del sistema o SUS [73], que sirve para tener una métrica general de la usabilidad del sistema probado.

Prueba de usabilidad	Descripción de la prueba
PU001.1. - Inicio de jornada.	El usuario debe iniciar sesión con unas credenciales dadas, entrar a la aplicación e iniciar un registro de trabajo.
PU002.1. - Comprensión de resumen.	El usuario debe encontrar las horas que lleva trabajadas durante el día y durante la semana, así como su porcentaje.
PU001.2. - Inicio de descanso.	El usuario debe ser capaz de iniciar un registro de descanso tras haber iniciado su jornada laboral.
PU002.2. - Comprensión de dashboard.	El usuario accederá a la sección *Summary* e irá explicando lo que entiende en cada elemento de la vista.
PU001.3. - Finalización de descanso.	El usuario debe ser capaz de reiniciar su jornada tras volver del descanso.
PU001.4. - Finalización de jornada.	El usuario debe ser capaz de finalizar su jornada, concretamente, finalizando un registro de trabajo.
PU003.1. - Modificación de registro.	El usuario deberá modificar la hora de finalización del registro de trabajo.
PU003.2. - Eliminación de registro.	El usuario deberá eliminar un registro de descanso.
PU004.1. - Creación de evento.	El usuario debe crear un evento en la vista de *My Calendar - Add New Event*.
PU004.2. - Visualización de eventos.	El usuario deberá explicar dónde se encuentra la información del evento creado.
PU004.3. - Visualización de calendario.	Se solicitará al usuario que visualice los eventos de sus compañeros en la sección de *Company Calendar*.
PU005.1. - Obtención de reportes.	Se solicitará al usuario administrador que exporte los reportes a PDF.
PU005.2. - Visualización de reportes.	Se solicitará al usuario administrador que filtre la tabla de reportes para encontrar un usuario en concreto.
PU006.1. - Creación de usuario.	El usuario administrador deberá crear un nuevo usuario.
PU006.2. - Eliminación de usuario.	El usuario administrador deberá eliminar el usuario creado.
PU007. - Cierre de sesión.	El usuario deberá cerrar sesión.

Resultados de las pruebas

Las pruebas fueron realizadas con dos usuarios técnicos y dos usuarios no técnicos. En concreto, los resultados obtenidos se presentan en la siguiente tabla:

Prueba	Éxito	Tiempo	Errores	Atributo us.	Resultado
PU001.1.	Sí.	5 - 20s.	No.	Eficiencia	Satisfactorio.
PU002.1.	Neutro.	20s. - 40s.	Interpretación de desplegable como botón Play/Pause.	F. aprendizaje.	Mejorable.
PU001.2.	Sí.	5 - 10s.	No.	Eficiencia	Satisfactorio.
PU002.2.	Neutro.	30s. - 1min.	Ver PU002.1.	F. aprendizaje	Mejorable.
PU001.3.	Sí.	5 - 10s.	No.	Eficiencia	Satisfactorio.
PU001.4.	Sí.	5 - 15s.	Intento de detención durante descanso.	Eficiencia	Satisfactorio.
PU003.1.	Bajo.	1m. - 1m. 30s.	Modificación errónea confusa.	F. aprendizaje	No satisfactorio.
PU003.2.	Sí.	5 - 10s.	No.	F. aprendizaje	Satisfactorio.
PU004.1.	Neutro.	1m. - 1m. 30s.	Usuario no técnico selecciona fechas desde el calendario.	F. aprendizaje	Mejorable.
PU004.2.	Sí.	10 - 30s.	No.	F. aprendizaje	Satisfactorio.
PU004.3.	Neutro.	10 - 30s.	Usuario no técnico: calendario compañía confuso.	F. aprendizaje	Mejorable.
PU005.1.	Sí.	10 - 20s.	No.	F. aprendizaje	Satisfactorio.
PU005.2.	Sí.	5 - 20s.	Intento de filtrado por horas (solo se admite id y nombre).	F. aprendizaje	Satisfactorio.
PU006.1.	Sí.	40s. - 1m.	No.	F. aprendizaje	Satisfactorio.
PU006.2.	Sí.	5 - 10s.	Posibilidad de eliminación de usuario único.	F. aprendizaje	Mejorable.
PU007.	Neutro.	10 - 20s.	Dificultad para encontrar botón.	Eficiencia	Mejorable.

Tras la realización de las pruebas de usabilidad y la obtención de las respuestas a la encuesta del SUS, se extraen una serie de conclusiones y puntos de mejora. Estos últimos detallados a continuación se solucionarán en parte tras la realización de las pruebas o se dejarán como puntos a mejorar en un futuro si se considera que no afectan a la funcionalidad principal de la aplicación y que solucionarlos podría suponer un aumento excesivo en el tiempo dedicado al proyecto.

En concreto, tras la realización de las pruebas de usabilidad con los usuarios técnicos se obtuvieron las siguientes conclusiones:

- Los pasos necesarios para realizar las acciones básicas no fueron en general muy diferentes a los mínimos necesarios, tampoco se detuvieron demasiado tiempo a la hora de pensar en cómo realizar una acción.
- Entre algunos de las posibles mejoras se encontraron:
  - Eliminar la posibilidad de borrar un usuario si este es el único que se encuentra en el sistema. Antes de eliminar un usuario completamente de la base de datos, preguntar al administrador si realmente quiere eliminarlo.
  - Si hay un registro en curso a la hora de cerrar sesión, preguntar al usuario si realmente quiere salir e informarle del estado de sus registros.
  - El porcentaje indicado sobre las horas semanales planeadas en la vista *Summary* es poco intuitivo al mostrarse sobre la suma de horas planeadas de los días que ya han pasado, en lugar de sobre la semana completa.
  - El borde que rodea el contador no siempre es suficiente para saber en qué registro se encuentra el usuario, podría ser necesario un título indicándolo.
  - Si el botón de Play/Pause se pulsa muchas veces seguidas puede suponer un fallo en la aplicación.
  - Al seleccionar las fechas de los eventos, podría mejorarse la respuesta de la aplicación estableciendo la fecha de finalización en un día después a la de inicio, si esta última ya ha sido seleccionada.
  - El botón para mostrar las vacaciones del departamento podría ser más adecuado a su funcionalidad si fuese un radio button.
  - En la tabla de gestión podría ser interesante mostrar también si el usuario es administrador y sus roles.
  - Sería útil indicar qué criterio de búsqueda se sigue en los filtros de *Management* y *Reports*.
  - Convendría añadir la posiblidad de limpiar el contenido del filtro con un símbolo o botón.
  - El contenido de la tabla que se muestra en *Reports* así como los registros de la vista *Timer* en el modo *list* pueden llegar a ser demasiados, por lo que debería reducirse el número que se muestra de los mismos.

- Una mejora interesante de cara al futuro sería añadir paginación en la vista de *Timer* o la posibilidad de acceder a una fecha concreta.
- Por último, el icono de ayuda podría mostrarse solo en la barra de la izquierda, ya que esta se mantiene constante en todas las secciones, en lugar de estar en diferentes zonas según la sección en la que se encuentre el usuario.

Además, tras la realización de las pruebas de usabilidad con los usuarios no técnicos se obtuvieron los siguientes puntos de mejora:

- La eliminación de usuarios desde *Management* debería tener una confirmación en dos pasos e impedirse en caso de que solo haya un usuario.
- Comunicar de algún modo que se ha modificado el registro.
- Los registros que indican que el usuario no estaba en la oficina podría ser innecesario mostrarlos.
- El tiempo total trabajado durante el día podría mostrarse al lado de la fecha.
- Señalar en la barra lateral qué sección es la actual.
- Cuando intentan añadir un evento intentan primero seleccionar las fechas en el calendario antes de pulsar el botón con el símbolo "+".
- En *Company Calendar* puede no resultar intuitivo qué fechas se están mostrando.
- Convendría una vista más simplificada de *Reports*.
- Añadir *feedback* cuando se añada un usuario a la base de datos y no permitir pulsar el botón de añadir si no están todos los campos del formulario cumplimentados.

Entre las mejoras que se decidió solucionar antes de la finalización del proyecto se encuentran:

- Cambios del texto de algunos botones para que su funcionalidad fuese más comprensible.
- Indicar criterio de búsqueda en los filtros.
- Limitar el número de registros mostrados en la vista de *Timer List*.

En cuanto a las respuestas obtenidas a las preguntas del SUS se aplica la siguiente fórmula [74]: 2,5x((I - 5) + (25 - P)), donde I representa la suma de las respuestas a las preguntas impares y P la suma de las respuestas pares; para obtener el resultado sobre la escala del SUS, representada en la **Figura 5.4**. Los resultados obtenidos para los usuarios técnicos son: 85 y 72,5 y para los usuarios no técnicos: 57,5 y 30. La media de estos resultados en conjunto sería de: 61,25.

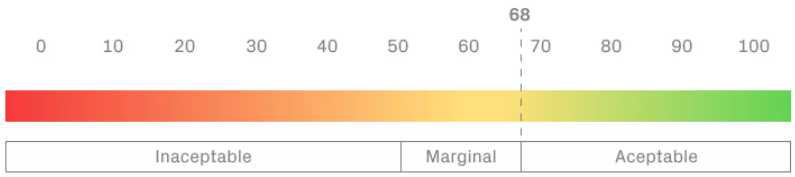


Figura 5.4: Representación gráfica del resultado del SUS

En general, los resultados positivos son mayores con usuarios técnicos que con usuarios no técnicos. Aunque estos no se podrían generalizar, dado que solamente se han realizado las pruebas con cuatro personas; es lógico pensar que una persona con conocimientos técnicos no considere necesaria la ayuda de otra persona con conocimientos técnicos para entender la aplicación o que vea el sistema menos complejo o más fácil de usar que un usuario no técnico.

Además, dado que el sistema va a ser de uso obligatorio y dentro del contexto de trabajo, no resulta difícil de entender que los usuarios en general no estén tan entusiasmados con su uso como podría darse con otra aplicación.

Sin embargo, dado que los resultados podrían ser mejores, sobre todo para los usuarios no técnicos; se plantea la necesidad de ahondar en las razones por las que los usuarios técnicos dan en general valoraciones mejores que los usuarios no técnicos, así como la de implementar mejoras en la interfaz.

Capítulo 6

Conclusiones y mejoras futuras

6.1. Conclusiones

Tras la finalización del proyecto, puede considerarse que los objetivos principales del mismo se han cumplido, pese a haberse prolongado la realización del mismo, quizá por ser estos demasiado ambiciosos.

En general, se ha implementado una aplicación web que permitirá a los usuarios de Biome Makers registrar y visualizar su jornada laboral, además de gestionar sus eventos; siendo este último un objetivo secundario respecto al registro de la jornada.

Ha sido posible además, realizar pruebas de usabilidad con usuarios finales de Biome Makers, aportando esto una mayor compresión del resultado final del proyecto.

En lo que respecta a los objetivos personales: se han adquirido o asentado conocimientos relativos a la Ingeniería del *software* y a la planificación de proyectos, se han reforzado conocimientos de análisis y gestión de bases de datos, en concreto con PSQL, se ha aprendido a crear una aplicación web desde cero con los *frameworks Laravel* y *Vue*, de los que inicialmente no se tenía conocimiento alguno, se ha profundizado en el diseño de interfaces y en la realización de pruebas, se ha realizado por primera vez un proyecto *software* completo, intentando seguir todas sus fases y aprendiendo de los errores cometidos durante las mismas.

6.2. Trabajo futuro

A lo largo del proyecto han surgido nuevas funcionalidades, que quedan fuera de los objetivos iniciales y que se plantearán en esta sección para su potencial extensión en un futuro. A estas hay que añadir algunas cuestiones que se han tenido en cuenta tras las pruebas de usabilidad.

En concreto, los puntos a mejorar serían los siguientes:

- Las *user stories* asociadas a la *initiative* cuyo objetivo sería notificar al usuario en determinadas situaciones.

- Mejorar el escalado de la aplicación para permitir su uso en otros dispositivos.
- Mejorar la eliminación de usuarios para que sea más segura.
- Mejorar aquellos detalles que hayan hecho la aplicación más difícil de entender a algunos usuarios, p.ej.: los colores asociados a cada tipo de registro.
- Permitir al usuario acceder a una fecha concreta para visualizar sus registros directamente, sin necesidad de desplazarse en orden o mostrar el tiempo total trabajado junto a la fecha.
- Crear una opción en la vista de *Reports* que permita ver la tabla de forma simplificada o agrupando por columnas.

Bibliografía

[1] Jefatura del Estado. Real decreto-ley 8/2019, de 8 de marzo, de medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo. https://www.boe.es/eli/es/rdl/2019/03/08/8, 2019. Accesed: 2021-04-05.

[2] César Laboy. Control horario para trabajadores: Conoce todo sobre la nueva ley. https://factorialhr.es/blog/nueva-ley-control-horario/, 2022. Accesed: 2021-02-13.

[3] Manual de usuario biotime 7.0. http://www.zksoftware.com.mx/assets/manual-usuario-biotime-7.0.pdf, 2018. Accesed: 2021-03-18.

[4] Software de control horario. top 5 software control horario 2020. https://programasb2b.com/top/software-control-horario/gratis, 2020. Accesed: 2021-04-28.

[5] Marina Camacho. 6 soluciones para hacer el registsro horario y fichar en el trabajo. https://factorialhr.es/blog/soluciones-fichar-en-el-trabajo/#que, 2021. Accesed: 2021-04-28.

[6] Top 5 software control horario gratis para tu empresa 2022. https://programasb2b.com/top/software-control-horario/gratis, 2022. Accesed: 2021-02-13.

[7] Absence. https://www.absence.io/es/. Accesed: 2021-02-21.

[8] Toggl. https://toggl.com/. Accesed: 2021-02-21.

[9] Wikipedia. Proceso para el desarrollo de software. https://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software. Accesed: 2021-04-25.

[10] B. Hughes. Software project management. en m. cotterell (ed.), software project management (5.a ed., pp. 82–86). mcgraw-hill education. WebPage, 2009. Accesed: 2021-04-24.

[11] IONOS. El modelo en cascada: desarrollo secuencial de software. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/. Accesed: 2021-04-26.

[12] Wikipedia. Desarrollo en cascada. https://es.wikipedia.org/wiki/Desarrollo_en_cascada. Accesed: 2021-04-26.

[13] IONOS. ¿qué es el modelo v? https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/modelo-v/. Accesed: 2021-04-26.

[14] IONOS. Modelo en espiral: el modelo para la gestión de riesgos en el desarrollo de software. https://www.ionos.es/startupguide/productividad/modelo-en-espiral/. Accesed: 2021-04-26.

[15] Manifiesto por el desarrollo Ágil de software. https://agilemanifesto.org/, 2001. Accesed: 2021-05-28.

[16] Wikipedia. Scrum (desarrollo de software). https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software). Accesed: 2021-04-28.

[17] Wikipedia. Desarrollo ágil de software. https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software. Accesed: 2021-04-28.

[18] Joel Francia Huambachano. ¿qué es scrum? https://www.scrum.org/resources/blog/que-es-scrum. Accesed: 2021-05-12.

[19] Atlassian. ¿qué es scrum? https://www.atlassian.com/es/agile/scrum. Accesed: 2021-05-12.

[20] Softeng. Proceso y roles de scrum. https://www.atlassian.com/es/agile/scrum. Accesed: 2021-05-12.

[21] Jeff Sutherland Ken Schwaber. La guía de scrum. https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf#zoom=100. Accesed: 2021-05-12.

[22] Atlassian. Historias, epics e iniciativas. https://www.atlassian.com/es/agile/project-management/epics-stories-themes. Accesed: 2021-05-12.

[23] IUSC. Conceptos y definiciones - camino crítico. https://www.iusc.es/recursos/gesproy/textos/03.02.09.htm. Accesed: 2021-05-20.

[24] Jaime de los Santos Francisco Allan Quijano. La gestión de riesgos en proyectos de desarrollo. https://blogs.iadb.org/efectividad-desarrollo/es/la-gestion-de-riesgos-en-proyectos-de-desarrollo/#:~:text=De%20acuerdo%20con%20la%20definici%C3%B3n,el%20costo%20y%20la%20calidad. Accesed: 2021-06-15.

[25] OBS. El plan de gestión de riesgos, ¿cómo se hace? https://www.obsbusiness.school/blog/el-plan-de-gestion-de-riesgos-como-se-hace. Accesed: 2021-06-17.

[26] Miguel Ángel Laguna Félix Prieto. Fundamentos de ingeniería del software: Requisitos. aulas infuva, 2018. Accesed: 2021-06-17.

[27] Miguel Ángel Laguna Félix Prieto. Fundamentos de ingeniería del software: Modelo de dominio. aulas infuva, 2018. Accesed: 2021-06-17.

[28] Universidad de Salamanca. Modelo de dominio. https://repositorio.grial.eu/bitstream/grial/1153/1/8.%20Modelo%20de%20dominio.pdf, 2018. Accesed: 2021-06-20.

[29] Miguel Ángel Laguna Félix Prieto. Fundamentos de ingeniería del software: Modelo de interacción. aulas infuva, 2018. Accesed: 2021-06-17.

[30] UVa. Arquitectura del software: Objetivos, definiciones, estilos arquitectónicos. patrones arquitectónicos: definición, catálogos. Campus UVa. Accesed: 2021-06-24.

[31] Biome makers. https://biomemakers.com/. Accessed: 2021-03-06.

[32] Jessica Colaluca. Design seeds. https://www.design-seeds.com/, 2009. Accesed: 2021-03-08.

[33] Wikipedia. Sitio web. https://es.wikipedia.org/wiki/Sitio_web. Accessed: 2022-01-13.

[34] Wikipedia. Desarrollo web. https://es.wikipedia.org/wiki/Desarrollo_web. Accessed: 2022-01-27.

[35] Laravel. Documentación de laravel - introducción. https://laravel.com/docs/8.x. Accessed: 2021-04-10.

[36] Laravel. Documentación de laravel - directory structure. https://laravel.com/docs/8.x/structure. Accessed: 2021-04-10.

[37] Laravel. Documentación de laravel - environment configuration. https://laravel.com/docs/8.x/configuration#environment-configuration. Accessed: 2021-04-10.

[38] Laravel. Documentación de laravel - enrutado. https://laravel.com/docs/8.x/routing. Accessed: 2021-04-10.

[39] Laravel. Documentación de laravel - vistas. https://laravel.com/docs/8.x/views. Accessed: 2021-04-10.

[40] Laravel. Documentación de laravel - eloquent. https://laravel.com/docs/8.x/eloquent. Accessed: 2021-04-10.

[41] Vue. Vue - guía. https://vuejs.org/v2/guide/. Accessed: 2022-01-22.

[42] Wikipedia. Vue.js. https://es.wikipedia.org/wiki/Vue.js. Accessed: 2022-01-27.

[43] Lenguajejs. Vue.js. https://lenguajejs.com/vuejs/introduccion/que-es-vue/. Accessed: 2022-01-29.

[44] Wikipedia. Base de datos. https://es.wikipedia.org/wiki/Base_de_datos. Accessed: 2022-01-22.

[45] UVa. Análisis y diseño de bases de datos: Introducción. Aulas infuva. Accessed: 2022-01-22.

[46] Wikipedia. Base de datos relacional. https://es.wikipedia.org/wiki/Base_de_datos_relacional. Accessed: 2022-01-22.

[47] Wikipedia. Sql. https://es.wikipedia.org/wiki/SQL. Accessed: 2022-01-22.

[48] Wikipedia. Postgresql. https://es.wikipedia.org/wiki/PostgreSQL. Accessed: 2022-01-22.

[49] Hostingpedia. Postgresql. https://hostingpedia.net/postgresql.html. Accessed: 2022-01-22.

[50] Wikipedia. Visual studio code. https://es.wikipedia.org/wiki/Visual_Studio_Code. Accessed: 2022-01-29.

[51] Visual Studio. Visual studio code. https://code.visualstudio.com/. Accessed: 2022-01-29.

[52] Wikipedia. Typescript. https://es.wikipedia.org/wiki/TypeScript. Accessed: 2022-01-29.

[53] MDN Web Docs Mozilla. Html. https://developer.mozilla.org/es/docs/Web/HTML. Accessed: 2022-01-29.

[54] Wikipedia. Sass. https://es.wikipedia.org/wiki/Sass. Accessed: 2022-01-29.

[55] Trello. https://recursostic.ucv.cl/wordpress/index.php/essential_grid/trello/#:~:text=TRELLO%20es%20un%20software%20de,como%20tambi%C3%A9n%20para%20trabajos%20colaborativos. Accessed: 2022-02-10.

[56] Qué es figma. https://3ymedia.school/que-es-figma/#:~:text=Figma%20es%20un%20editor%20de,como%20en%20Mac%20o%20Linux. Accessed: 2022-02-10.

[57] Wikipedia. Github. https://es.wikipedia.org/wiki/GitHub. Accessed: 2022-02-20.

[58] Xataka. Qué es github y qué es lo que le ofrece a los desarrolladores. https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores. Accessed: 2022-02-20.

[59] Laravel. Writting commands. https://laravel.com/docs/9.x/artisan#writing-commands. Accessed: 2022-02-28.

[60] Wikipedia. Pruebas de *software*. https://es.wikipedia.org/wiki/Pruebas_de_software. Accessed: 2022-02-20.

[61] Wikipedia. Iso/iec 9126. https://es.wikipedia.org/wiki/ISO/IEC_9126. Accessed: 2022-02-20.

[62] Atlassian. ¿qué son las pruebas de software? https://www.atlassian.com/es/continuous-delivery/software-testing. Accessed: 2021-12-04.

[63] Pruebas de caja negra: Fase de pruebas software. https://educandocontic.com/pruebas-caja-negra/. Accessed: 2022-02-20.

[64] medium. La famosa pirámide de cohn y la dura realidad. https://medium.com/@wc.testing.qa/la-famosa-pir%C3%A1mide-de-cohn-y-la-dura-realidad-e1250dfbe5f3, 2018. Accessed: 2022-02-20.

[65] La pirámide de cohn. https://nicopaez.gitbook.io/test-automation/clasificacion-de-pruebas/piramide_de_pruebas, 2021. Accessed: 2022-02-20.

[66] Sofia Palamarchuk. La pirámide de cohn. https://cl.abstracta.us/blog/piramide-de-automatizacion/. Accessed: 2022-02-20.

[67] Glenn Lee. Tipos de pruebas de software: diferencias y ejemplos. https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/. Accessed: 2022-02-20.

[68] Wikipedia. Prueba de usablidad. https://es.wikipedia.org/wiki/Prueba_de_usabilidad. Accessed: 2022-03-02.

[69] Laravel. Testing: Getting started. https://laravel.com/docs/8.x/testing. Accessed: 2022-02-28.

[70] Laravel. Database testing. https://laravel.com/docs/9.x/database-testing. Accessed: 2022-02-28.

[71] Laravel. Http tests. https://laravel.com/docs/7.x/http-tests#assert-see. Accessed: 2022-02-28.

[72] Laraveltip. Haciendo pruebas automatizadas en laravel [parte i]. https://www.laraveltip.com/haciendo-pruebas-automatizadas-en-laravel/. Accessed: 2022-02-28.

[73] usability.gov. System usability scale (sus). https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. Accessed: 2022-02-28.

[74] ¿cómo medir la usabilidad con un sus? https://www.uifrommars.com/como-medir-usabilidad-que-es-sus/. Accessed: 2022-03-02.

[75] ObjectRocket. Laravel and postgresql web app part 1. https://kb.objectrocket.com/postgresql/laravel-and-postgresql-web-app-part-1-885, 2019. Accessed: 2021-04-10.

[76] ObjectRocket. Laravel and postgresql web app part 2. https://kb.objectrocket.com/postgresql/laravel-and-postgresql-web-app-part-2-886, 2019. Accessed: 2021-04-10.

[77] ObjectRocket. Laravel and postgresql web app part 3. https://kb.objectrocket.com/postgresql/laravel-and-postgresql-web-app-part-3-887, 2019. Accessed: 2021-04-10.

[78] ObjectRocket. Laravel and postgresql web app part 7. https://kb.objectrocket.com/postgresql/laravel-and-postgresql-web-app-part-7-1099, 2019. Accessed: 2021-04-10.

[79] Mark Drake. Cómo instalar y utilizar postgresql en ubuntu 20.04. https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04-es, 2020. Accessed: 2021-04-25.

[80] Wikipedia. Scrum. https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)#Historia, 2021. Accessed: 2021-05-13.

[81] Marina Camacho. 6 soluciones para hacer el registro horario y fichar en el trabajo. https://factorialhr.es/blog/soluciones-fichar-en-el-trabajo/#que, 2021. Accessed: 2021-04-12.

[82] César Laboy. Control horario para trabajadores: Conoce todo sobre la nueva ley. https://factorialhr.es/blog/nueva-ley-control-horario/, 2021. Accessed: 2021-04-12.

[83] Wikipedia. Pruebas de software. https://es.wikipedia.org/wiki/Pruebas_de_software. Accessed: 2021-12-04.

[84] Laravel. Documentación de laravel - lifecycle. https://laravel.com/docs/8.x/lifecycle. Accessed: 2021-04-10.

[85] Laravel. Documentación de laravel - service container. https://laravel.com/docs/8.x/container. Accessed: 2021-04-10.

[86] Laravel. Documentación de laravel - facades. https://laravel.com/docs/8.x/facades. Accessed: 2021-04-10.

[87] Rafalinux. Escribir código en latex. http://www.rafalinux.com/?p=599. Accessed: 2021-01-22.

[88] Wikipedia. Identificador de recursos uniforme. https://es.wikipedia.org/wiki/Identificador_de_recursos_uniforme. Accessed: 2021-01-22.

[89] Wikipedia. Asignación objeto-relacional. https://es.wikipedia.org/wiki/Asignaci%C3%B3n_objeto-relacional. Accessed: 2021-01-22.

[90] Wikipedia. Phpstorm. https://en.wikipedia.org/wiki/PhpStorm. Accessed: 2022-01-29.

[91] Jetbrains. Phpstorm. https://www.jetbrains.com/es-es/phpstorm/. Accessed: 2022-01-29.

[92] PHP. ¿qué es php? https://www.php.net/manual/es/intro-whatis.php. Accessed: 2022-01-29.

[93] Wikipedia. Css. https://es.wikipedia.org/wiki/CSS. Accessed: 2022-01-29.

[94] Usability. System usability scale. https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. Accessed: 2021-12-04.

Apéndice A

Instrucciones para pruebas de usabilidad

A continuación, con el fin de comprobar si la aplicación web que se le mostrará cumple con ciertas características, se le va a solicitar que intente llevar a cabo una serie de acciones. Si es posible, será de utilidad que explique o describa todo aquello que vaya pensando en el proceso que siga para llevar a cabo dichas acciones. Aun cuando actúe como se le indica, cabe la posibilidad de que sea interrumpido durante el desarrollo de la prueba para realizarle preguntas sobre lo que acaba de hacer o lo que está viendo.

Antes de esto, se deberá haber creado al menos un usuario administrador y un usuario no administrador.

Las acciones que se le van a proponer son las siguientes:

1. Acaba de llegar a la oficina y su *email* y contraseña son las credenciales del usuario que le haya sido asignado, dado que va a comenzar su jornada, ¿cómo iniciaría sesión y procedería a crear el registro de sus horas de trabajo?

2. ¿Sabrías decirnos cuántas horas deberías llevar trabajadas a día de hoy y cuántas llevas trabajadas? ¿Y el porcentaje de las mismas?

3. Llevas un rato trabajando y te apetece descansar y tomar un café, ¿cómo introducirías un descanso en tu jornada de trabajo?

4. Accede a la vista de resumen, por orden, ve enumerando los elementos en los que te fijes y explica qué información extraes de estos.

5. Tu descanso ha terminado, ¿cómo reiniciarías tus horas de trabajo?

6. Tras terminar tus horas de trabajo, ¿qué harías para que quede constancia en la aplicación de que ya has terminado?

7. Te has confundido y deseas modificar la hora de finalización de este último registro, ¿cómo lo harías?

8. ¿Y si quisieras eliminar el registro de descanso?
9. ¿Cómo crearías un evento?
10. Tras mostrar el evento, ¿puedes indicar dónde se muestra la información del mismo?
11. ¿Y si quieres ver los eventos de tus compañeros? Para mostrar solo los de aquellos que tienen cargos similares al tuyo, ¿qué harías?

Las siguientes acciones aparecerán solo si la cuenta es de administrador:

1. Acceda a los reportes de los usuarios e indique cómo exportaría un .PDF con todos los datos. ¿Y si lo que quiere es ver uno en concreto?
2. Para añadir un nuevo usuario, ¿qué haría? ¿y para eliminarlo?

Ya ha terminado casi, ¿cómo cerraría sesión?

Finalmente, valora del 1 al 5 las siguientes preguntas según esté más o menos en desacuerdo (siendo 1 nada de acuerdo y 5 totalmente de acuerdo):

1. Me gustaría usar este sistema con frecuencia.
2. El sistema me ha parecido innecesariamente complejo.
3. Creo que el sistema ha sido fácil de usar.
4. Creo que necesitaría la ayuda de una persona con conocimientos técnicos para ser capaz de usar este sistema.
5. Creo que las diferentes funciones del sistema estaban bien integradas.
6. Me parece que hay demasiadas inconsistencias en este sistema.
7. Creo que la mayoría de personas aprenderían a usar este sistema muy rápido.
8. El sistema me pareció de uso incómodo.
9. Me sentí seguro usando el sistema.
10. Necesité aprender muchas cosas antes de que pudiera aprender a usar este sistema.

Resultados de las pruebas de usabilidad

Prueba realizada con usuarios técnicos

Respuestas del primer usuario técnico a las preguntas del SUS. Valora del 1 al 5 las siguientes preguntas según esté más o menos en desacuerdo (siendo 1 nada de acuerdo y 5 totalmente de acuerdo).

1. Me gustaría usar este sistema con frecuencia. **3**
2. El sistema me ha parecido innecesariamente complejo. **1**
3. Creo que el sistema ha sido fácil de usar. **4**
4. Creo que necesitaría la ayuda de una persona con conocimientos técnicos para ser capaz de usar este sistema. **1**
5. Creo que las diferentes funciones del sistema estaban bien integradas. **3**
6. Me parece que hay demasiadas inconsistencias en este sistema. **1**
7. Creo que la mayoría de personas aprenderían a usar este sistema muy rápido. **5**
8. El sistema me pareció de uso incómodo. **1**
9. Me sentí seguro usando el sistema. **4**
10. Necesité aprender muchas cosas antes de que pudiera aprender a usar este sistema. **1**

Resultado final: 2,5 x((19 - 5) + (25 - 5)) = 2,5 x 34 = 85.

Respuestas del segundo usuario técnico a las preguntas del SUS. Valora del 1 al 5 las siguientes preguntas según esté más o menos en desacuerdo (siendo 1 nada de acuerdo y 5 totalmente de acuerdo):

1. Me gustaría usar este sistema con frecuencia. **3**
2. El sistema me ha parecido innecesariamente complejo. **2**
3. Creo que el sistema ha sido fácil de usar. **4**
4. Creo que necesitaría la ayuda de una persona con conocimientos técnicos para ser capaz de usar este sistema. **1**
5. Creo que las diferentes funciones del sistema estaban bien integradas. **3**
6. Me parece que hay demasiadas inconsistencias en este sistema. **2**

7. Creo que la mayoría de personas aprenderían a usar este sistema muy rápido. **4**
8. El sistema me pareció de uso incómodo. **3**
9. Me sentí seguro usando el sistema. **4**
10. Necesité aprender muchas cosas antes de que pudiera aprender a usar este sistema. **1**

Resultado final: 2,5 x((18 - 5) + (25 - 9)) = 2,5 x 29 = 72,5.

Prueba realizada con usuarios no técnicos

Respuestas del primer usuario no técnico a las preguntas del SUS. Valora del 1 al 5 las siguientes preguntas según esté más o menos en desacuerdo (siendo 1 nada de acuerdo y 5 totalmente de acuerdo):

1. Me gustaría usar este sistema con frecuencia. **2**
2. El sistema me ha parecido innecesariamente complejo. **2**
3. Creo que el sistema ha sido fácil de usar. **3**
4. Creo que necesitaría la ayuda de una persona con conocimientos técnicos para ser capaz de usar este sistema. **2**
5. Creo que las diferentes funciones del sistema estaban bien integradas. **4**
6. Me parece que hay demasiadas inconsistencias en este sistema. **2**
7. Creo que la mayoría de personas aprenderían a usar este sistema muy rápido. **2**
8. El sistema me pareció de uso incómodo. **2**
9. Me sentí seguro usando el sistema. **4**
10. Necesité aprender muchas cosas antes de que pudiera aprender a usar este sistema. **4**

Resultado final: 2,5 x((15 - 5) + (25 - 12)) = 2,5 x 23 = 57,5.

Respuestas del segundo usuario no técnico a las preguntas del SUS. Valora del 1 al 5 las siguientes preguntas según esté más o menos en desacuerdo (siendo 1 nada de acuerdo y 5 totalmente de acuerdo):

1. Me gustaría usar este sistema con frecuencia. **2**
2. El sistema me ha parecido innecesariamente complejo. **3**

3. Creo que el sistema ha sido fácil de usar. **3**
4. Creo que necesitaría la ayuda de una persona con conocimientos técnicos para ser capaz de usar este sistema. **4**
5. Creo que las diferentes funciones del sistema estaban bien integradas. **3**
6. Me parece que hay demasiadas inconsistencias en este sistema. **4**
7. Creo que la mayoría de personas aprenderían a usar este sistema muy rápido. **2**
8. El sistema me pareció de uso incómodo. **3**
9. Me sentí seguro usando el sistema. **1**
10. Necesité aprender muchas cosas antes de que pudiera aprender a usar este sistema. **3**

Resultado final: 2,5 x((15 - 11) + (25 - 17)) = 2,5 x 12 = 30.

Apéndice B

Manual de Usuario

En esta sección se realizará un sencillo tutorial sobre el funcionamiento de la aplicación, que permite generar los registros de la jornada laboral del usuario correspondiente, visualizar un resumen de los mismos, así como gestionar los eventos y vacaciones. Además, presenta una serie de funcionalidades extra para la administración de los usuarios del sistema que solo podrán ser utilizadas por un usuario con permisos de administrador.

Para crear el primer usuario, dado que no se permitirá el auto-registro, se facilitará un comando con el que crear un usuario administrador. Para mostrar todos los comandos disponibles, se ejecuta la orden:

```
php artisan list
```

Esto muestra una serie de comandos entre los que se muestra el comando de la **Figura B.1**.

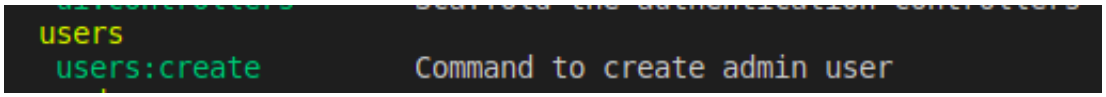


Figura B.1: Comando *users:create*

Para ejecutar este comando, se ejecuta la orden

```
php artisan users:create
```

de forma que se irán preguntando por terminal los datos del usuario administrador, como se muestra en la **Figura B.2**.

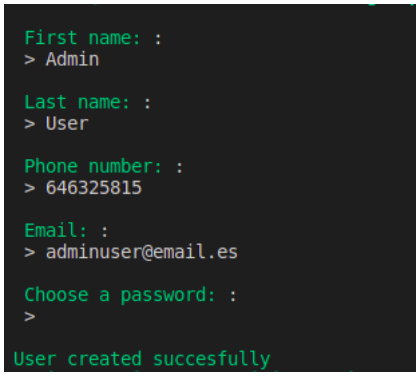


Figura B.2: Comando *users:create*

Lo primero que encontrarás como usuario, tanto si eres administrador, como si no, es la pantalla de ***Login***. En ella se te solicitará tu email y contraseña correspondientes, como se puede ver en la Figura B.3. Además, en caso de no saber cómo realizar alguna acción o querer conocer todas las posibilidades existentes en cada vista, encontrarás un icono de ayuda, que en el *Login* se encuentra en la parte inferior derecha de la pantalla y al pulsarlo mostrará un tutorial sobre la pantalla en la que te encuentras.

Figura B.3: Login

Una vez has iniciado sesión, se te mostrará la pantalla correspondiente al **Resumen *(Summary)***. En el centro de la misma encontrarás un resumen de las horas trabajadas durante el día y la semana actuales en porcentaje, junto con una tabla que mostrará las fechas de tus eventos próximos.

En la parte superior de la pantalla, podrás encontrar una barra superior extensible donde verás

un resumen en horas y minutos del tiempo trabajado durante el día, la semana o el mes, así como las horas pendientes acumuladas y las horas extra acumuladas hasta la fecha actual.

Junto a este resumen encontrarás un contador del tiempo acumulado. Este estará a cero y bordeado con un color amarillento si el estado actual es "Fuera de la oficina". Si estás trabajando se mostrará el tiempo que llevas acumulado trabajando desde el último inicio de jornada y el contador estará bordeado en verde. Por último, si te encuentras en un descanso, se mostrará bordeado en naranja y acumulará el tiempo desde el inicio de descanso.

Al lado de este contador se encuentra el botón principal destinado al registro de la jornada. Su parte central permite iniciar un registro de trabajo o pausar la jornada, iniciando un registro de descanso, mientras que el botón superior permite detener la jornada si se encuentra en un registro de trabajo y el inferior permite cerrar la sesión.

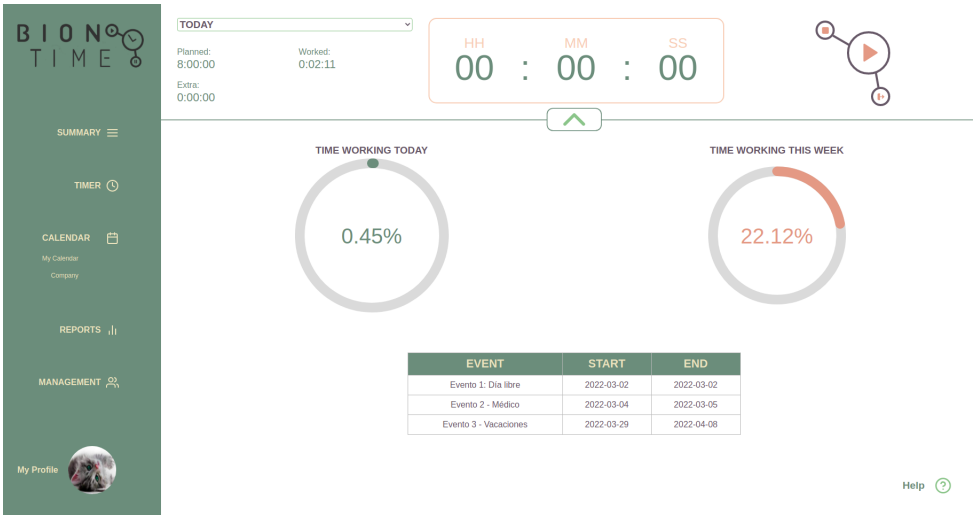


Figura B.4: Summary

Mediante los enlaces situados en la barra lateral podemos acceder a esta y al resto de secciones de la aplicación. Las secciones de *Reports* y *Management* solo serán visibles para el administrador, un usuario sin estos permisos solo verá las secciones mostradas en la Figura B.5. La siguiente al Resumen es la sección de ***Timer***, donde encontrarás de nuevo la barra superior de gestión de jornada. El resto del contenido permite visualizar y gestionar los registros ya creados.

Esta sección presenta tres tipos de visualización, seleccionables a través del desplegable situado en la parte superior derecha. La primera forma de visualización es la de tipo Lista, que se muestra en la Figura B.6. En ella, encontrarás una barra separada en tres, con los colores correspondientes a los estados: 'trabajando", "en un descanso" y "fuera de la oficina". Los colores de esta barra ocuparán mayor o menor espacio de la misma, dependiendo de qué tipo de registro haya ocupado un mayor número de horas en porcentaje sobre el total.

Debajo de esta barra encontramos los registros separados por fecha, de diferentes colores, dependiendo de a qué tipo de registro correspondan. Se mostrarán dentro de estos, las horas de

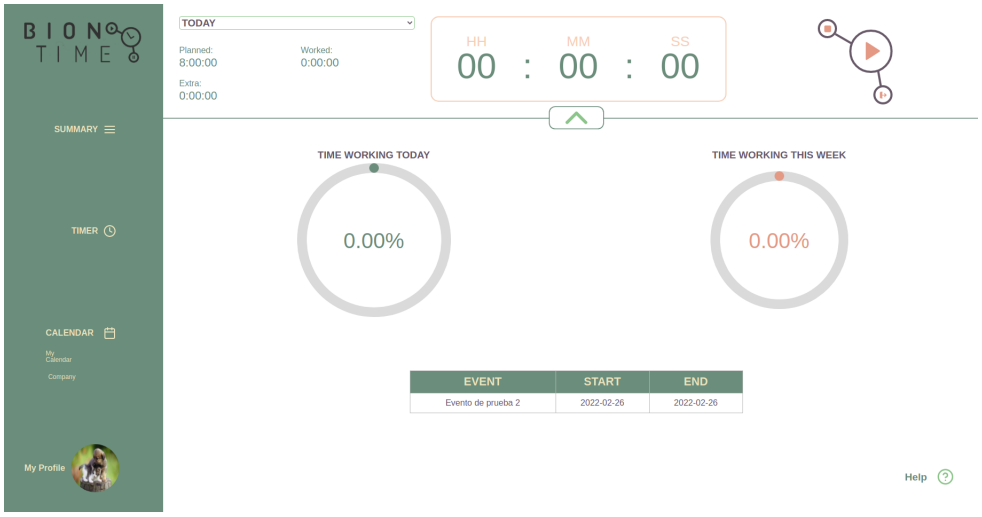


Figura B.5: Vista para usuario no administrador

comienzo y finalización del registro y el tiempo total comprendido entre ambas. Además, sobre estos registros se mostrará la opción de modificarlos, que convertirá las horas de comienzo y finalización en campos editables; y al lado de cada registro la opción de eliminarlo (que podrá hacerse o no, dependiendo de las características del registro: se podrá eliminar si se trata de un descanso o de un registro de trabajo aislado).

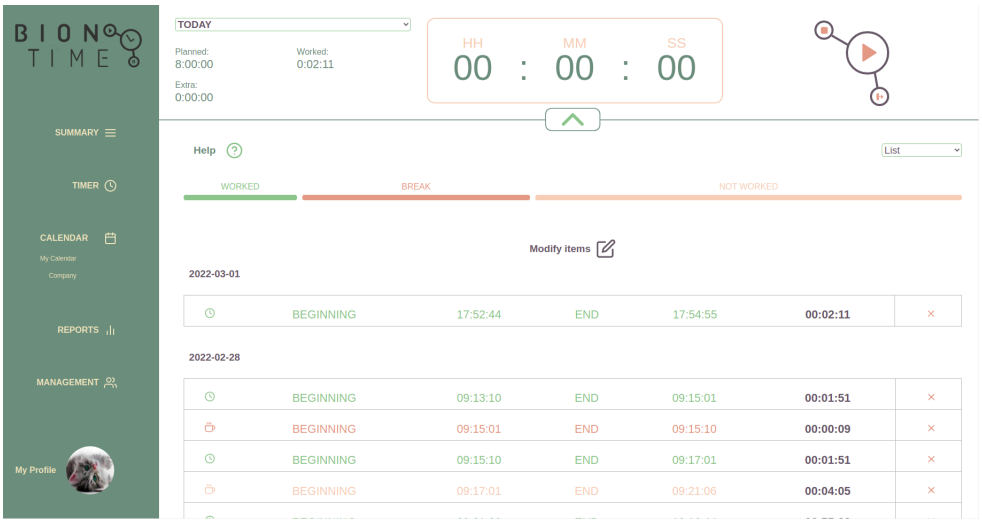


Figura B.6: Timer

Si seleccionamos la segunda opción del desplegable, nos aparece algo similar a lo mostrado en la Figura B.7, donde se verán los registros de un solo día. El valor de la fecha podrá modificarse

utilizando las flechas situadas en la parte superior izquierda, siempre y cuando la fecha que se desea seleccionar sea igual o anterior a la actual.

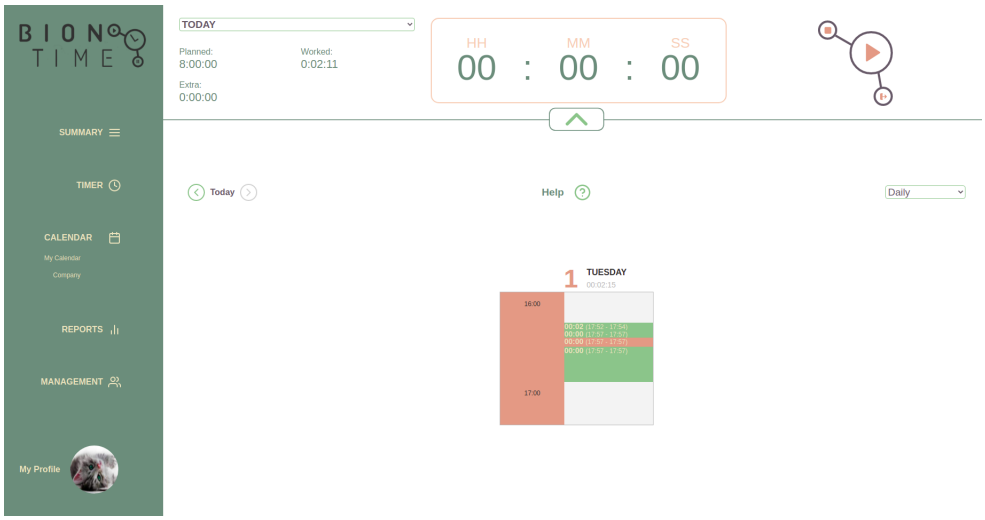


Figura B.7: Timer Daily View

En la tercera opción se muestran los registros correspondientes a la fecha actual junto con los de fechas anteriores dentro de la semana correspondiente. Es decir, si hoy es miércoles se mostrarán los registros del lunes, del martes y del miércoles de esta semana. Al igual que con la opción anterior, podemos desplazarnos por las semanas, siempre que las fechas que solicitemos sean iguales o anteriores a la actual.

A la siguiente opción a seleccionar en la barra lateral se puede acceder pulsando tanto en la palabra **"CALENDAR"** como en la palabra **"My Calendar"**. Esta mostrará, siguiendo la estética de las dos secciones anteriores, una barra superior donde encontrarás una leyenda indicando el color de los eventos según su tipo y a la derecha del título, una lista de los últimos eventos añadidos a tu calendario.

Bajo esta barra superior, encontrarás un calendario donde verás las fechas en las que tienes algún evento en otro color. Si pulsas sobre ellas, podrás ver más información sobre los mismos. Además, si eres un usuario administrador, te aparecerá bajo el día correspondiente un puntito azul indicando que algún otro usuario tiene un evento en esa fecha. Al igual que para visualizar la información de tus eventos, es suficiente con pulsar sobre la fecha para poder verlo en más detalle.

En la esquina inferior izquierda encontrarás una *checkbox* para indicar si deseas o no que otros usuarios puedan visualizar tus eventos. Y, por último, en la esquina inferior derecha, podrás ver un botón que, al pulsarlo, mostrará un formulario para añadir nuevos eventos. En este formulario se te solicitará el nombre del evento, su tipo y fechas de inicio y finalización. Si estas fechas están ya reservadas por otro de tus compañeros de un rol parecido se te avisará. Lo mismo ocurrirá si las fechas son incompatibles entre sí o si son anteriores a la fecha actual.

En la sección de ***Company Calendar***, seleccionable justo bajo la de "My Calendar" encon-

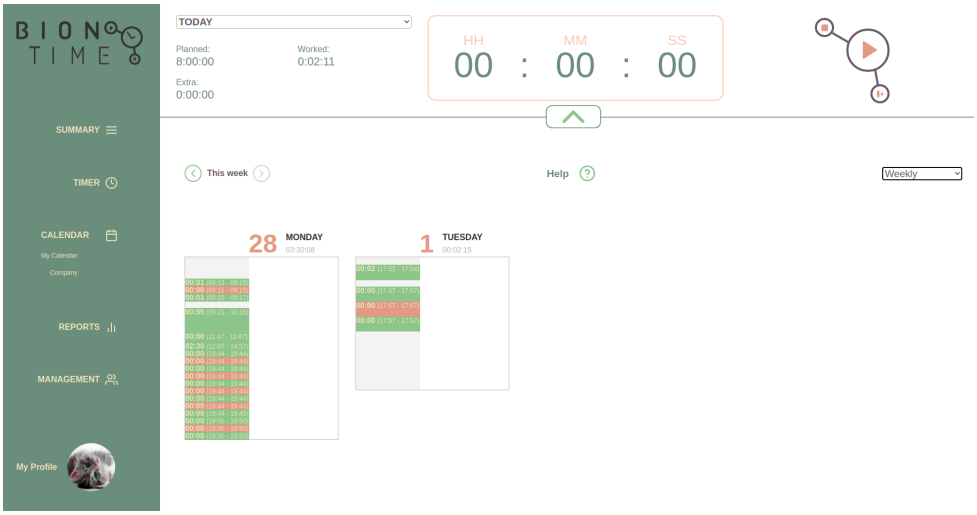


Figura B.8: Timer Weekly View

trarás un horario semanal que mostrará los eventos de otros compañeros, siempre que estos hayan seleccionado que permiten que sus eventos sean visibles. Al igual que en la sección anterior, encontrarás una leyenda de colores para diferenciar los tipos de eventos. Además, se te permitirá filtrar por función o rol pulsando en el botón "Show Department Holidays", donde verás solo los eventos que puedan afectar a tu trabajo.

Además, si eres un usuario administrador, podrás acceder a las dos secciones que se explican a continuación: La sección de ***Reports*** y la sección de ***Management***.

En la sección de ***Reports*** se mostrarán los registros de los diferentes usuarios del sistema o el mensaje "No records" en caso de que no tengan. Aparecerán en una tabla con los siguientes datos: el identificador y nombre y apellidos del usuario en cuestión, la fecha y las horas de comienzo y finalización de cada registro. Además, se mostrará el tiempo total que ha estado activo, si el tiempo corresponde a un registro de trabajo o el tiempo total que ha estado parado si corresponde a un registro de descanso, en las dos últimas columnas.

Sobre esta tabla se mostrarán dos opciones, un filtro por si quieres visualizar un usuario o identificador concretos y un desplegable que te permitirá exportar la tabla completa a formato .PDF o .CSV si pulsas en el botón "Download File".

En la sección de ***Management*** se muestra una tabla similar a la anterior, con el mismo filtro, pero esta vez encontrarás el identificador, nombre y apellidos del usuario, email y teléfono. Al lado de cada registro verás dos iconos. El primero te permitirá modificar el usuario y el segundo, eliminarlo.

Bajo la tabla encontrarás un botón, "Add New User" que te redirigirá a la página que se muestra en la Figura B.14. En ella, tendrás que rellenar un formulario con los datos del usuario (no se puede utilizar un email que ya esté en la tabla) indicando si es un administrador, los roles que le corresponden y su imagen de usuario.

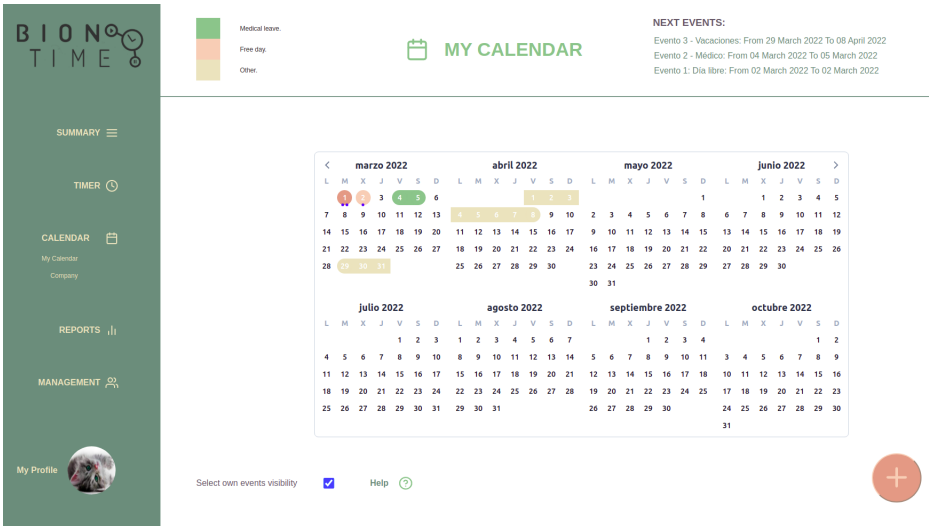


Figura B.9: My Calendar

Tras completarlo, será suficiente con pulsar el botón “Add”.

Un formulario similar al de “Add New User” es el que se muestra al pulsar el icono de modificar junto al registro de cada usuario. En este caso, el formulario aparecerá parcialmente rellenado con los datos actuales del usuario. El email no podrá modificarse al ser el dato identificativo del usuario y las contraseñas deberán ser iguales para que se realice dicha modificación.

Por último, tanto el usuario administrador como el no administrador podrán ver sus datos en la sección de ***My Profile***, donde además, encontrarán la opción de cerrar la sesión actual.

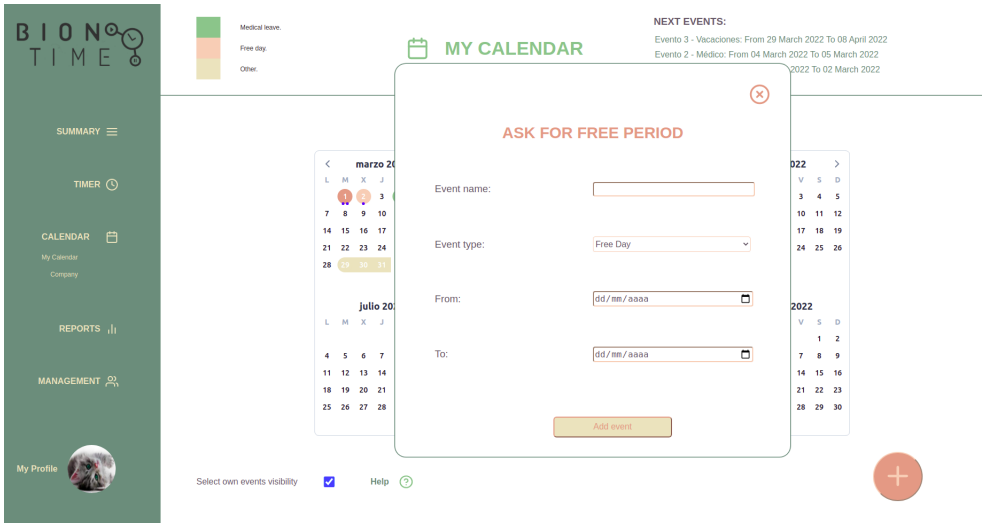


Figura B.10: Add Event

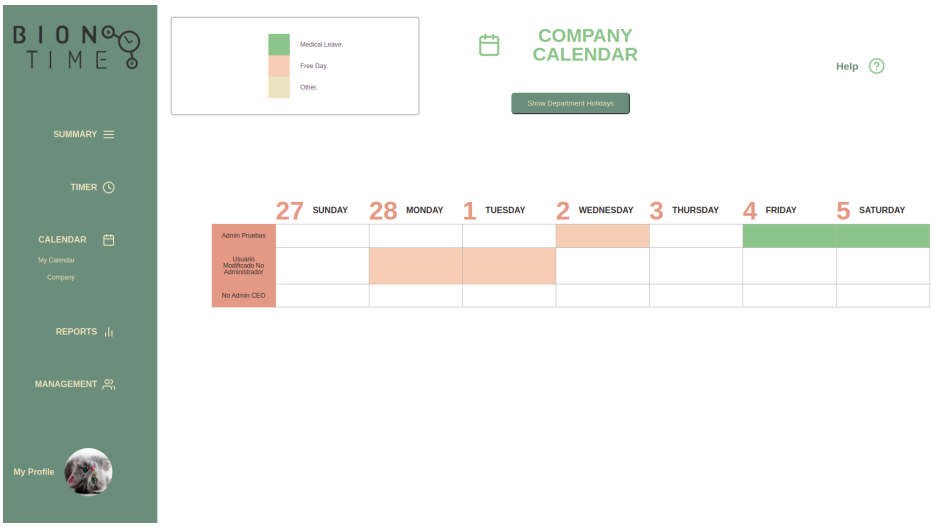


Figura B.11: Company Calendar

Figura B.12: Reports

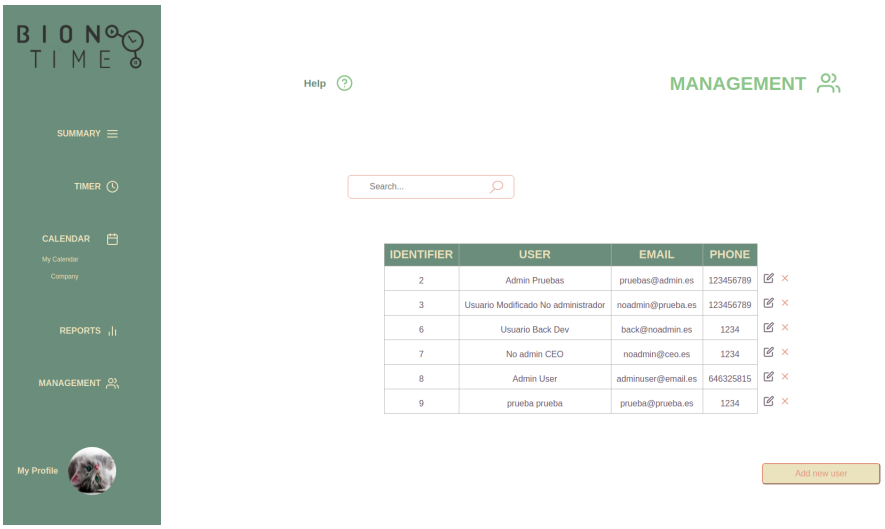


Figura B.13: Management

Figura B.14: Add New User

Figura B.15: Modify User

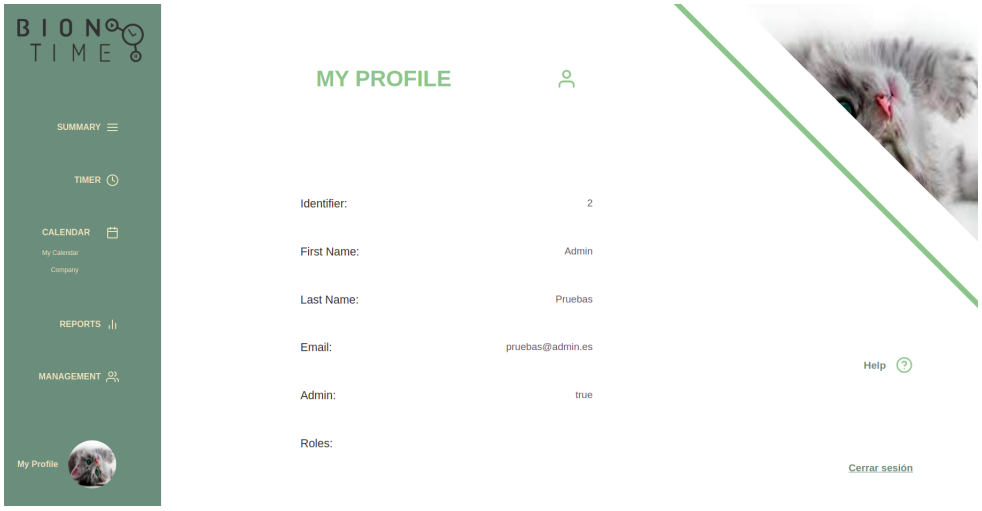


Figura B.16: My Profile