UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Máster Universitario en Ingeniería Web

Trabajo Fin de Máster

Aplicación Web para la gestión de correo postal

Autor

Blanca Fanny Castaño Caceres

Tutor

Santiago Alonso Villaverde

enero de 2024

Contenido

1 Resumen

Este proyecto Fin de Máster surge de la necesidad de crear una solución que permita gestionar el estado de los sobres o paquetes de una empresa de entrega de correo postal. Cada sobre o paquete tiene asignado un numero de guía, siendo una guía un documento que se agrega a dicho sobre con un número de identificación único para poder hacer el seguimiento de dicho sobre o paquete.

La gestión e información del estado de un sobre consiste en agregar manualmente el número de guía del sobre cuando entra a la empresa en un fichero de registros de entrada llamado “Cargue y fecha actual” siendo creado uno por día. El siguiente paso es asignar dicho número de guía a un empleado dependiendo de la zona. Por último, cuando el sobre es entregado se agrega a otro fichero llamado “Descargue y fecha actual” o si es devuelto por algún motivo se agrega al fichero “Devolución y fecha actual”, estos ficheros también son creados uno por día. Este procedimiento requiere mucho tiempo y dedicación por parte del director de operaciones.

Lo que se persigue con la aplicación a desarrollar en este proyecto Fin de Máster es automatizar esta tediosa labor permitiendo al director de operaciones acceder con un usuario y contraseña a través de la aplicación para poder gestionar por medio del sistema todas las operaciones necesarias para que el estado de un envío este siempre actualizado y accesible.

En esta aplicación el director de operaciones podrá agregar envíos, modificar su estado, consultar dicho estado y obtener listados de guías con parámetros de búsqueda diferentes, estos parámetros pueden ser por estado de la guía, fecha entrada, fecha cargue (asignación a empleado) y fecha de entrega. De esta forma, podrá saber en estado se encuentra un envío.

2 Abstract

This Master's Thesis arises from the need to create a solution that allows managing the status of envelopes or packages from a postal mail delivery company.

Each envelope or package is assigned a tracking number, a tracking number being a document that is added to said envelope with a unique identification number to be able to track said envelope or package. The management and information on the status of an envelope consists of manually adding the tracking number of the envelope when it enters the company in an entry record file called “Load and current date,” one being created per day. The next step is to assign said guide number to an employee depending on the area. Finally, when the envelope is delivered, it is added to another file called “Unload and current date” or if it is returned for some reason it is added to the file “Return and current date”, these files are also created one per day. This procedure requires a lot of time and dedication on the part of the operations director.

What is sought with the application developed in this TFM is to automate this tedious work of checks and notifications, allowing the operations director to access with a username and password through the application to be able to manage through the system all the operations necessary to that the status of a shipment is always updated and accessible.

In this application the director of operations will be able to add shipments, modify their status, consult said status and obtain lists of guides with different search parameters, these parameters can be by status of the guide, entry date, assignment date (assignment to employee) and end Date. This way, it will be able to know the status of a shipment.

3 Objetivos

El principal objetivo de este proyecto Fin de Máster es el desarrollo de una aplicación web en la que toda la gestión del correo postal permita la gestión de los envíos de manera integral. Para ello, se utilizarán los conocimientos y herramientas adquiridos durante este máster, así como los conocimientos adquiridos en el ámbito laboral.

Se han marcado los siguientes objetivos específicos:

- Identificar los pasos necesarios dentro de la metodología Agile para la correcta gestión del desarrollo de una aplicación web.
- Aplicar una de las arquitecturas aprendidas en el desarrollo de una aplicación web.
- Crear un sistema web que permita gestionar el estado de los envíos postales de manera integral.
- Utilizar las técnicas adecuadas y recomendadas por la Ingeniería del Software para garantizar, en la medida de lo posible, la calidad de los productos obtenidos.
- Usar las tecnologías Angular en lado del cliente y Spring en el lado del servidor para el desarrollo del sistema y establecer un sistema de integración continua.

4 Introducción

Este proyecto fin de máster intenta cubrir la necesidad existente de gestionar el estado de los envíos postales.

El proceso de registro de entrada de un sobre o paquete se realiza cuando el sobre llega a la empresa, este sobre tiene adjunta una guía que es un documento con un número de identificador único. Este número de guía es agregado a un fichero Excel generado llamado “Cargue + fecha” que se genera uno cada día. El número guía agregado se ubica debajo de una celda con texto “nombre de zona” siendo la zona el lugar al que va dirigido el paquete y pudiendo haber más de una zona en un fichero. Una vez el sobre es entregado, se agrega nuevamente el numero de la guía en un fichero Excel llamado “Descargue + fecha” debajo de otra celda con el texto “nombre de zona”. Cuando un sobre no puede ser entregado por ejemplo porque la dirección es errada o algún otro motivo, este número de guía es agregado también al fichero Excel llamado “Devoluciones + fecha” debajo de una celda que contiene el texto “nombre de zona”. Los ficheros de descargue y devolución también son generados uno por día.

En dicha hoja de cálculo se puede apreciar por bloques los números de guías y si se necesita saber el estado en el que se encuentra un envío es necesario buscar por varias hojas de “Descargue” ya que puede ser de ese día o de posteriores, ya que puede ocurrir que se haya recibido y puesto en estado “Cargue” una fecha y se haya entregado y puesto en estado “Descargue” o “Devolución” en otra, pudiéndose encontrar el estado final de dicha guía en una variedad de ficheros posibles.

Este método de gestión de correo supone demasiado tiempo de dedicación para el director de operaciones, persona que actualmente lleva a cabo este proceso. Las comparaciones y búsquedas necesarias de estos ficheros deben ser gestionadas manualmente.

En este proyecto fin de Máster se pretende crear una aplicación Web basada en Angular en lado del cliente y Spring en el lado del servidor. La persistencia se realizará en una base de datos SQL, para agilizar y facilitar la tarea de tal forma que el director de operaciones pueda satisfacer la necesidad de gestionar los estados y búsquedas de un sobre o paquete. Este proyecto será gestionado con la metodología Ágil Scrum ya que es una de las vistas en este Máster y en la que se tiene experiencia por cuestiones laborales.

En la aplicación a desarrollar en este Proyecto fin de Máster, el director de operaciones podrá acceder con un usuario y contraseña para gestionar el estado de los envíos postales, siendo la principal ventaja que el director de operaciones no tendrá que agregar varias veces la misma guía para gestionar el estado en el que se encuentra sino que será el sistema quien una vez agregada una guía se encargue de almacenarla una única vez en la base de datos y solo se deba modificar el estado de ésta cuando sea necesario a través de la aplicación, evitando así el manejo de multitud de ficheros para obtener el estado de una guía. La aplicación también se encargará de generar listados por estado de guía, por fecha, por zona o por empleado.

5 Metodología de desarrollo

La metodología de software se refiere a las formas en las cuales se realiza la planificación para el diseño del software, básicamente con el objetivo de mejorar, optimizar procesos y ofrecer una mejor calidad.

Antes que nada veamos ¿qué es un método? y también ¿qué es una metodología?

Un **Método** es un conjunto de herramientas, técnicas y procesos que facilitan la obtención de un objetivo; una **Metodología** hace cierto énfasis al entorno en el cual se plantea y estructura el desarrollo de un sistema; es importante que dependiendo del tipo de software que se vaya a desarrollar, se identifique la metodología para el diseño de software idónea, ya que no todos los sistemas de la información, son compatibles con todas las metodologías, pues el ciclo de vida del software puede ser variable; una metodología de desarrollo de software, consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo.

Las metodologías vistas en este Máster son:

5.1 RUP. Rational Unified Process

Se organiza en torno a fases y flujos de trabajo, empleando un enfoque iterativo. Sus principales características son:

5.1.1 Dirigido por casos de uso

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo como se muestra en la ilustración 1.

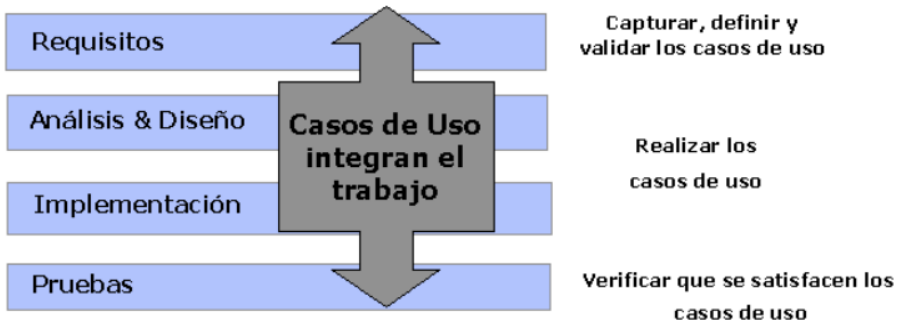


Ilustración 1. Rup, Casos de uso integran el trabajo

5.1.2 Proceso centrado en la arquitectura

RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura, los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

En la Ilustración 2 se ilustra la evolución de la arquitectura durante las fases de RUP. Se tiene una arquitectura más robusta en las fases finales del proyecto. En las fases iniciales lo que se hace es ir consolidando la arquitectura por medio de baselines y se va modificando dependiendo de las necesidades del proyecto.

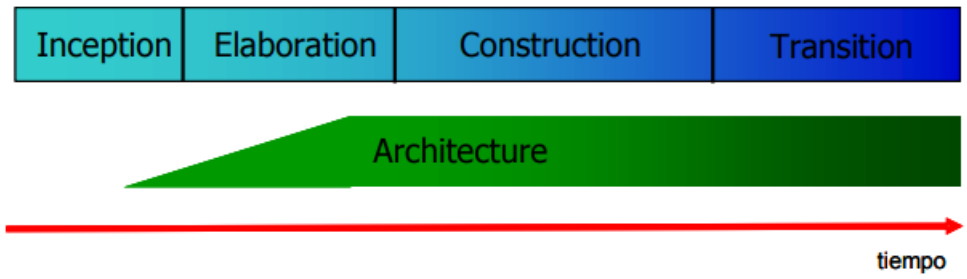


Ilustración 2. Evolución de la arquitectura del sistema

5.1.3 Proceso iterativo e incremental

La estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada parte. Cada parte se puede ver como una iteración del cual se obtiene un incremento que produce un crecimiento en el producto. Una iteración puede realizarse por medio de una cascada como se muestra en la Ilustración 3.

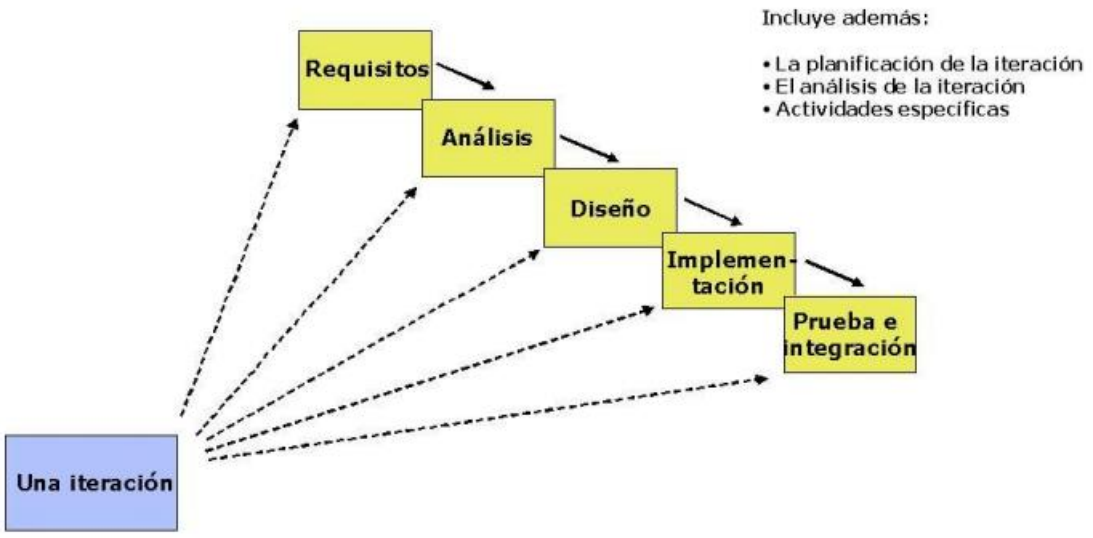


Ilustración 3. Iteración RUP

Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. En cada iteración se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Se realizan iteraciones hasta que se termine la implementación de la nueva versión del producto.

5.1.4 Estructura del proceso

El proceso puede ser descrito en dos ejes :

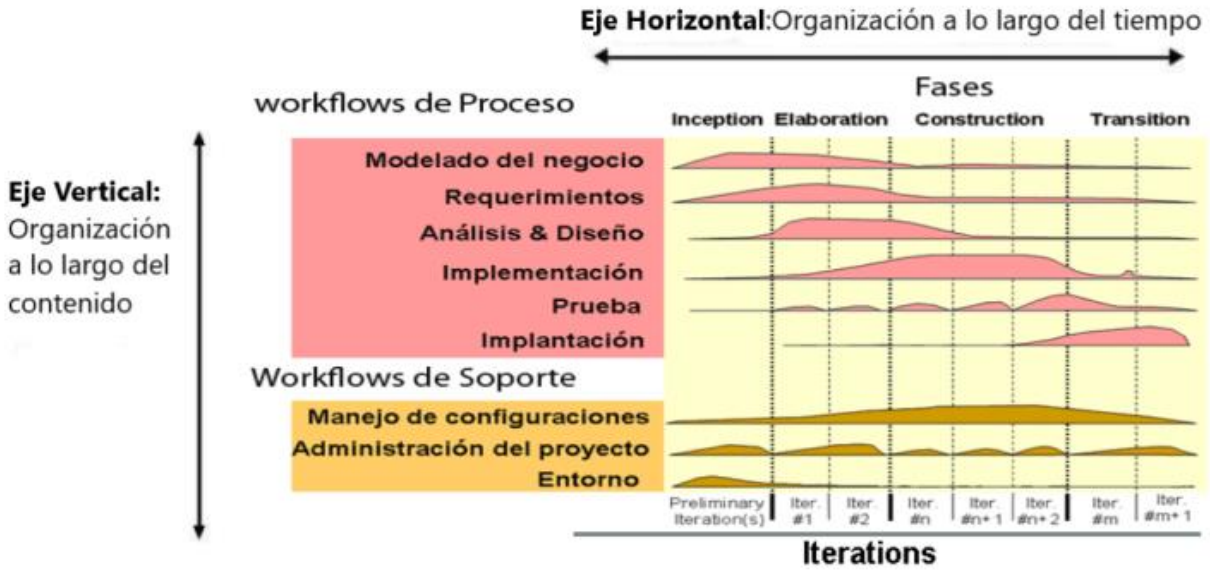


Ilustración 4. Estructura de Rup

Eje horizontal: Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos. Se puede observar en la Ilustración 5 que RUP consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase se subdivide a la vez en iteraciones.

La duración y esfuerzo dedicado en cada fase es variable dependiendo de las características del proyecto. Sin embargo, la Ilustración 5 se aprecian los porcentajes frecuentes al respecto.

	Inicio	Elaboración	Construcción	Transición
Esfuerzo	5 %	20 %	65 %	10%
Tiempo Dedicado	10 %	30 %	50 %	10%

Ilustración 5. Rup distribución típica de tiempo y esfuerzo

Eje vertical: Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

Un proceso de desarrollo de software es el que define quién hace qué, cómo y cuándo. RUP define cuatro elementos:

- los roles, que responden a la pregunta ¿Quién?
- las actividades que responden a la pregunta ¿Cómo?
- los productos, que responden a la pregunta ¿Qué?
- los flujos de trabajo de las disciplinas que responde a la pregunta ¿Cuándo?

5.2 Agile

Estas metodologías se centran en la integración de componentes o en la capacidad de adaptación a los requerimientos que surgen en el desarrollo, de allí su nombre metodologías ágiles.

Las metodologías ágiles son una serie de técnicas para la gestión de proyectos que han surgido como contraposición a los métodos tradicionales.

Todas las metodologías que se consideran ágiles cumplen con el manifiesto ágil que no es más que una serie de principios que se agrupan en 4 valores:

1. Los individuos y su interacción, por encima de los procesos y las herramientas.
2. El software que funciona, frente a la documentación exhaustiva.
3. La colaboración con el cliente, por encima de la negociación contractual.
4. La respuesta al cambio, por encima del seguimiento de un plan.

Las metodologías ágiles mejoran la satisfacción del cliente dado que se involucrará y comprometerá a lo largo del proyecto. En cada etapa del desarrollo se informará al cliente sobre los progresos de este. De ese modo, el cliente puede sumar su experiencia para optimizar las características del producto final. Se pueden evitar así numerosos malentendidos dado que el cliente poseerá en todo momento una completa visión del estado del producto.

Así mismo, mejora la motivación e implicación del equipo de desarrollo. Pero esta mejora no es casual: las metodologías ágiles permiten a todos los miembros del equipo conocer el estado del proyecto en cualquier momento. Los compromisos son negociados y aceptados por todos los miembros del equipo y las ideas de cualquiera de sus integrantes son tenidas en cuenta.

En las metodologías ágiles se trabaja realizando entregas parciales pero funcionales del producto. De ese modo, es posible entregar en el menor intervalo de tiempo posible una versión funcional del producto. Gracias a las entregas parciales (centradas en entregar en primer lugar aquellas funcionalidades que en verdad aportan valor) y al contar con la implicación del cliente será posible eliminar aquellas características innecesarias del producto.

Las metodologías ágiles permiten mejorar la calidad del producto. La continúa interacción entre los desarrolladores y con el cliente tiene como objetivo asegurar que el producto final sea exactamente lo que el cliente quiere y necesita.

Dentro de la metodología ágil existe una primera parte a realizar que otorga la visión del proyecto que se va a desarrollar. Al comienzo de un proyecto existen un conjunto de tareas más obvias, pero igualmente importantes de definir como son:

- El alcance, entendiéndolo como lo que se va a hacer y lo que no se va a hacer.
- Poner de acuerdo con todos los interesados (stakeholders) en la visión conjunta del proyecto, lo que se pretende conseguir y lo que no se pretende conseguir.
- Qué riesgos existen que puedan poner en peligro el proyecto.
- Qué alternativa de solución se propone para llevarlo a cabo.
- Cuanto y hasta cuando llevara hacerlo.

Estas decisiones, aunque pueden sufrir alguna variación en el tiempo, deben ser abordadas al comienzo de un proyecto para evitar posibles repercusiones posteriores. El desarrollo del proceso de toma de estas decisiones las veremos más adelante en el

Road Map

5.2.1 Scrum

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

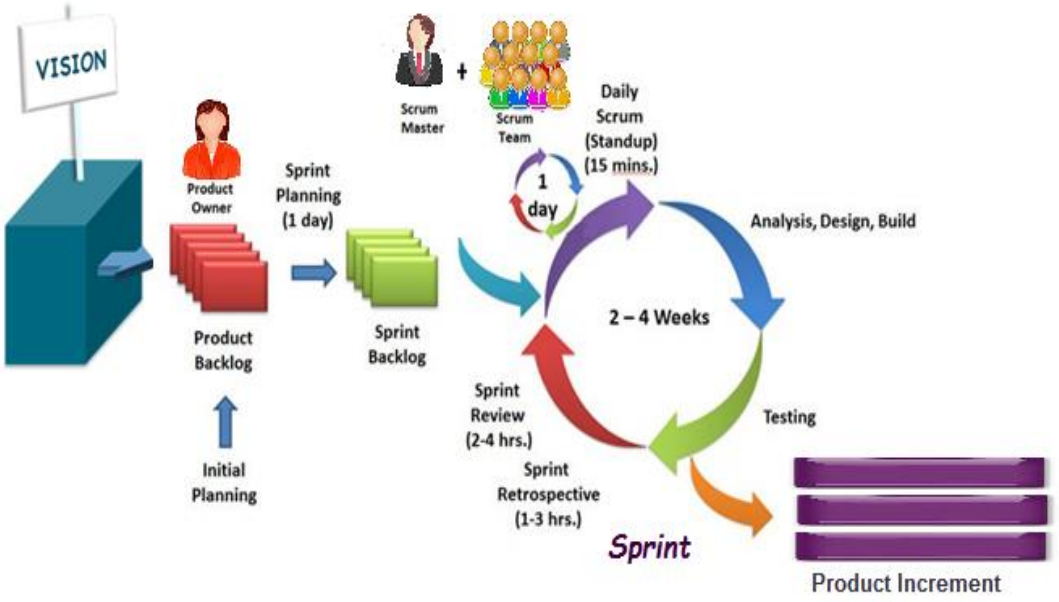


Ilustración 6. Scrum

Equipos que conforman la metodología Scrum:

- ***Product Owner:*** Es quién da respuesta a la pregunta "¿Qué hay que hacer?". Si se trata de tener un líder de proyecto, entonces el Product Owner lo será. Básicamente son los ojos del cliente, será la persona encargada del proyecto y de comprobar que se lleve a cabo de tal forma que cumpla las expectativas de lo que se espera.
- ***Scrum Master:*** Es el encargado de determinar la solución de los problemas y cada complicación que suceda. para cada reunión realizada, siempre debe estar

un líder, en este caso el Scrum Master será el líder de cada una de las reuniones y ayudará en los problemas que hayan surgido. Será básicamente como un "facilitador" el cual minimizará obstáculos, sin embargo no los omitirá.

- ***Scrum Team***: Es el núcleo de la metodología Scrum, pues es el equipo de desarrollo, encargado de la codificación del software y de cumplir los objetivos o metas propuestas por el Product Owner.
- ***Cliente***: El cliente tiene la capacidad para influir en el proceso, debido a que siempre estará empapado de él, ya sea que proponga nuevas ideas o bien haciendo algún tipo de comentario.

Procesos con los cuales funciona la metodología:

- ***Product Backlog***: Es una lista de las funcionalidades del producto a desarrollar. Este debe ser elaborado por el Product Owner, debe estar ordenado de acuerdo con las prioridades del sistema de más a menos, con la idea de que las cosas con mayor prioridad sean las que se realicen antes de cualquier cosa.
- ***Sprint Backlog:*** Consiste en seleccionar algunos de los puntos escritos en el Product Backlog, los cuales procederán a ser realizados, en este punto el Sprint Backlog tiene como requisito marcar el tiempo en que se llevará a cabo el Sprint.
- ***Sprint Planning Meeting***: Es una reunión que se realiza para definir plazos y procesos a efectuarse para el proyecto establecido en el Product Backlog. Cada Sprint, se compone de diversas características (features), que no son otra cosa más que procesos o subprocesos que se deben realizar, puede ser la creación de un logo, la gestión de contenido, el diseño visual, etc.
- ***Daily Scrum o Stand-up Meeting***: Aquí lo que se hace son reuniones diarias de aproximadamente 15 minutos mientras se está llevando a cabo un Sprint, para responder las siguientes preguntas: ¿Que hice ayer?, ¿Qué voy a hacer hoy, ¿Qué ayuda necesito? Aquí entra en función el Scrum Master.
- ***Sprint Review***: Una reseña de lo que fue el Sprint, consiste en la revisión del Sprint terminado y para este punto ya tendría que haber algo que mostrarle al cliente, algo realmente visual o tangible para que se pueda analizar un cierto avance.
- ***Sprint Retrospective***: Permite al equipo analizar los objetivos cumplidos, si se cometieron errores, visualizarlos y tratar de no cometerlos nuevamente más adelante; este proceso sirve para la implementación de mejoras.

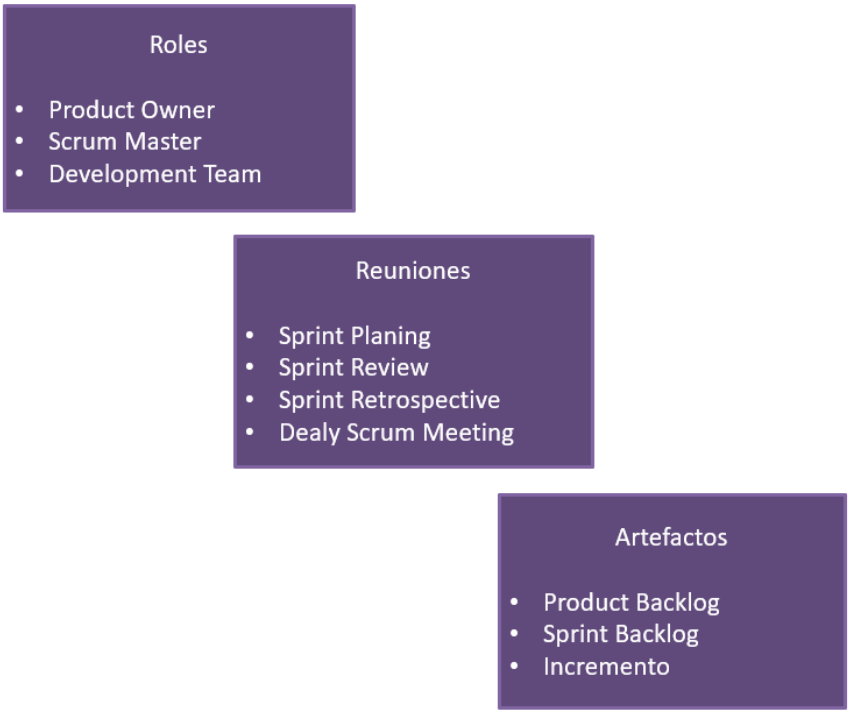


Ilustración 7. Componentes de Scrum

5.3 Metodología elegida para la aplicación

Una vez analizadas las dos metodologías se ha decidido utilizar para la gestión del proyecto la metodología Scrum por diferentes motivos, he aquí una lista de estos:

- Se adapta mejor al proyecto ya que se pueden hacer mejoras rápidas y organizaciones que no dependen de una fecha límite.
- Los requerimientos son flexibles y/o el ámbito puede tender al cambio .
- Se tiene experiencia por motivos laborales.

6 Visión de la aplicación

Se han establecido diferentes etapas en el desarrollo del proyecto:

1. Una inicial de conceptualización usando Inception DeckInception Deck.
2. Una posterior de desglose, definición y desarrollo del proyecto, utilizando el Road Map para obtener una mejor visión de los requisitos, funcionalidades y mapa de desarrollo esperados por el cliente.

Se hablará del planteamiento y evolución del proyecto usando Scrum. Se hablará de artefactos y eventos de Scrum que se tratarán de trasladar en apartados posteriores a herramientas. Cada apartado se inicia con una breve descripción de lo que consiste y posteriormente, la información concreta para nuestro proyecto.

El flujo de desarrollo de Scrum es incremental e iterativo, por lo que al inicio del proyecto es habitual tener solo una idea más bien vaga de lo que se quiere. Se darán tiempos de entrega aproximados, los cuales podrán variar según las complicaciones que puedan surgir en cada Release.

6.1 Inception Deck

Inception Deck o simplemente *Inception*, es un conjunto de dinámicas orientadas a enfocar a todas las personas involucradas en un proyecto hacia un mismo objetivo, ayudando a identificar los riesgos más evidentes, poniendo en común las expectativas de todos los stakeholders; teniendo como objetivo reducir las incertidumbres que puedan surgir permitiendo de esta manera reducir la falta de consenso, problemas de comunicación y la ambigüedad en el proyecto.

Estás dinámicas permiten la identificación de funcionalidades, roadmapping y definición de una arquitectura de alto nivel. El inception deck contiene en total 10 preguntas; éste se describe originalmente como un “deck” (baraja) porque no es obligatorio pasar por todas las preguntas (aunque sí recomendable). Así, dependiendo del contexto, se pueden sustituir unas actividades por otras que persigan los mismos objetivos o incluso obviarlas si esos objetivos ya se han conseguido previamente.

La duración del Inception Deck debe ser entre dos días a una semana para una planificación de hasta 6 meses

Los apartados desde el 1 hasta el 5 se centran en actividades dinámicas que están orientadas a discutir de manera creativa sobre el producto que se quiere construir, centradas todas en el QUÉ y obviando siempre el CÓMO.

6.1.1 ¿Por qué estamos aquí?

El objetivo de esta actividad es entender cuál es el producto que queremos construir. Esto nos va a servir para poder tomar mejores y más claras decisiones con respecto a lo que queremos, de esta manera se aportarán mejores soluciones sobre los conflictos que puedan surgir en el futuro ya que se tiene una visión clara de QUÉ es lo estamos construyendo.

La gestión e información del estado de un sobre o paquete en una empresa de correo postal se realiza de manera manual ingresando los datos en un fichero. El principal inconveniente que se encuentra el director de operaciones al gestionar el estado de un paquete es que debe buscar en un fichero Excel por número de guía y ver si lo

encuentra, ya que el número de guía y su estado es agregado en diferentes ficheros según el estado en el que se encuentre.

El proceso de registro de entrada de un sobre o paquete se realiza cuando el sobre llega a la empresa, este sobre tiene adjunta una guía que es un documento con un número de identificador único. Este número de guía es agregado a un fichero Excel generado llamado “Cargue + fecha” que se genera uno cada día. La guía agregada se ubica debajo de una celda con texto “nombre de zona” siendo la zona el lugar al que va dirigido el paquete y pudiendo haber más de una zona en un fichero. Una vez el sobre es entregado, se agrega nuevamente el numero de la guía en un fichero Excel llamado “Descargue + fecha” debajo de otra celda con el texto “nombre de zona”. Cuando un sobre no puede ser entregado por ejemplo porque la dirección es errada o algún otro motivo, este número de guía es agregado también al fichero Excel llamado “Devoluciones + fecha” debajo de una celda que contiene el texto “nombre de zona”. Los ficheros de descargue y devolución también son generados uno por día.

En dicha hoja de cálculo se puede apreciar por bloques los números de guías y si se necesita saber el estado en el que se encuentra un envío es necesario buscar por varias hojas de “Descargue” ya que puede ser de ese día o de posteriores, pudiendo ocurrir que se haya recibido y puesto en estado “Cargue” una fecha y se haya entregado y puesto en estado “Descargue” o “Devolución” otra día pudiéndose encontrar el estado final de dicha guía en una variedad de ficheros posibles.

Este método de gestión de correo supone demasiado tiempo de dedicación para el director de operaciones, persona que actualmente lleva a cabo este proceso. Las comparaciones y búsquedas necesarias de estos ficheros deben ser gestionadas manualmente.

En la aplicación a desarrollar en este Proyecto fin de Master, el director de operaciones podrá acceder con un usuario y contraseña para gestionar el estado de los envíos postales de manera integral, siendo la principal ventaja que el director de operaciones no tendrá que agregar varias veces la misma guía para gestionar el estado en el que se encuentra sino que será el sistema quien una vez agregada una guía se encargue de almacenarla una única vez en la base de datos y solo se deba modificar el estado de ésta cuando sea necesario a través de la aplicación, evitando así el manejo de multitud de ficheros para obtener el estado de una guía. La aplicación también se encargará de generar listados por estado de guía, por fecha, por zona o por empleado.

6.1.2 Conversación de ascensor

El objetivo de esta actividad es conseguir explicar en aproximadamente 40 segundos (tiempo medio que puede durar una conversación en un ascensor) nuestro producto.

Veamos ahora la “conversación de ascensor” para nuestra aplicación:

La aplicación de gestión de correo postal es una aplicación Web para una empresa de correo postal que satisface la necesidad de gestionar los estados de una sobre o paquete. En esta aplicación el director de operaciones podrá acceder con un usuario y

contraseña, para agregar un sobre o paquete al sistema a través del número que tiene asignado en un documento llamado guía. Una vez agregado el número de guía al sistema el director de operaciones puede modificar el estado en el que se encuentra el paquete a través de un desplegable que le permite elegir los estados disponibles. Finalmente el director de operaciones puede obtener listados de guías por estado, fecha entrada en la empresa, fecha de asignación a un empleado, fecha de entrega y por el numero identificativo de la guía.

6.1.3 Diseña tu caja

En este apartado el objetivo es construir una caja de producto como si la ofreciéramos en un centro comercial, junto a sus slogans principales. Diseñar una caja de producto es un concepto relacionado con marketing que nos invita a imaginar la caja de un producto con los mensajes e imágenes necesarias como si nuestro producto estuviera empaquetado y fuera a estar expuesto de forma que resulte lo más atractivo posible para el cliente.

Veamos ahora el “diseño de la caja” para nuestra aplicación:

Beneficios:

- **Acceso desde cualquier lugar**: Permite acceder a un usuario, independientemente de su ubicación a la aplicación.
- **Automatización**: Todos los ingresos que se realizan en la empresa tienen la opción de modificación de estado, gracias a esto las consultas realizadas sobre los envíos se obtienen con el estado correcto y actualizado.

Ilustración 8. Diseña tu caja

6.1.4 No list

El objetivo de esta actividad es hacer una lista:

IN SCOPE: Características que entran en el ámbito de nuestro producto, pueden se funcionalidades de alto nivel u objetivos generales.

OUT OF SCOPE: Lo que no será nuestro producto. Contiene características que no vamos a hacer o que son parte de otra fase futura.

UNRESOLVED: Las características que no tenemos claro que entren en el ámbito de nuestro producto; son aquellas características sobre las que se debe tomar una decisión.

De esta manera obtendremos una visión más clara de lo que será nuestro producto, lo que NO será y aquello que aún está indefinido.

Veamos ahora "el Not List" para nuestra aplicación:

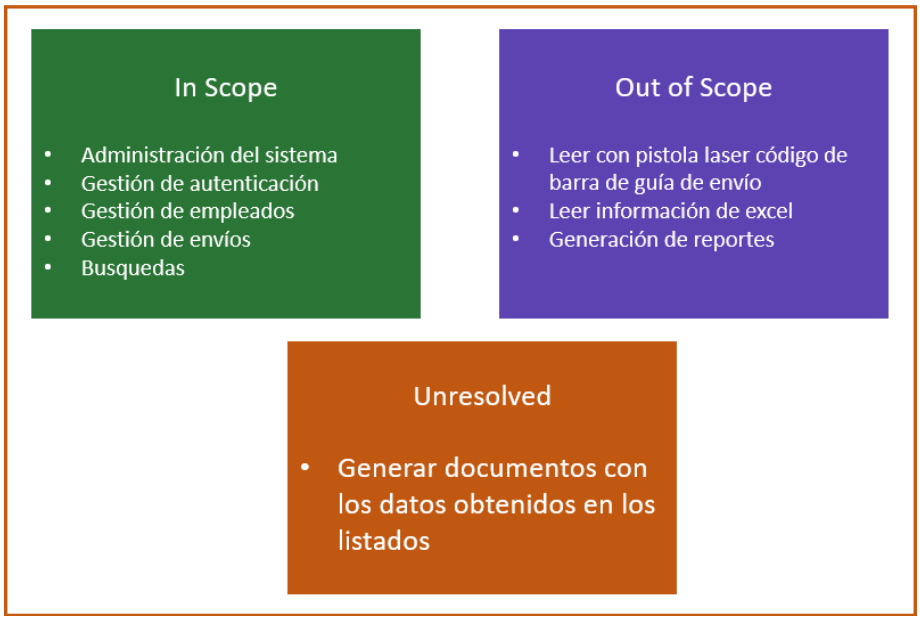


Ilustración 9. Not list

6.1.5 Conoce a tus vecinos

El objetivo de esta actividad es identificar a todos aquellos que no son parte de nuestro equipo de trabajo pero que están involucrados en nuestro proyecto de alguna manera, para así poder tener una relación de confianza con todas aquellas personas involucradas en nuestro proyecto.

Veamos ahora “Conoce a tus vecinos” para nuestra aplicación:

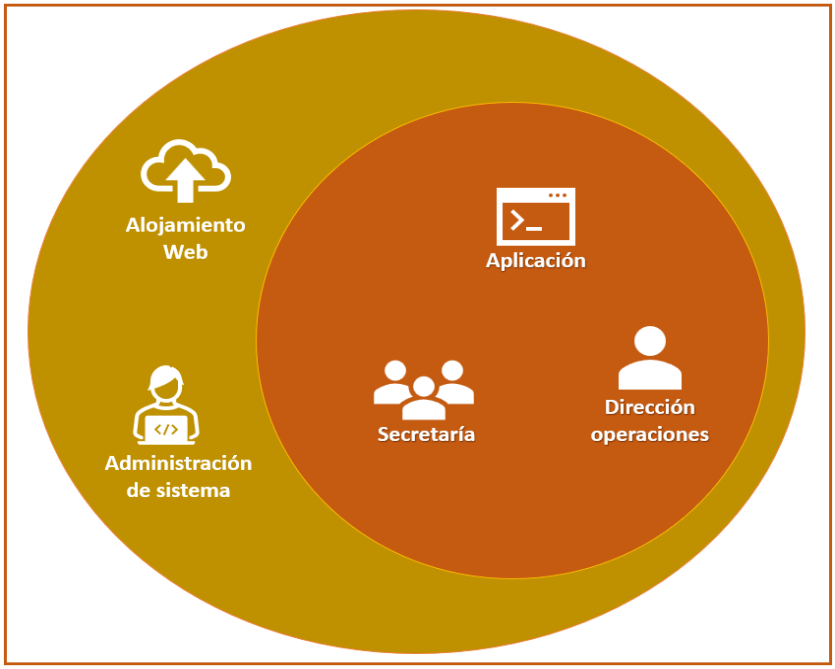


Ilustración 10. Conoce a tus vecinos

6.1.6 Visualizar la Solución

Esta actividad se centra en el CÓMO realizaremos nuestro producto. El objetivo de esta actividad es obtener una visualización global de la arquitectura que utilizaremos para el desarrollo y de los escenarios en los que estará comprendo el uso de nuestro producto.

Veamos ahora “Visualiza la Solución” para nuestra aplicación en los escenarios:

- Cliente
- Servidor

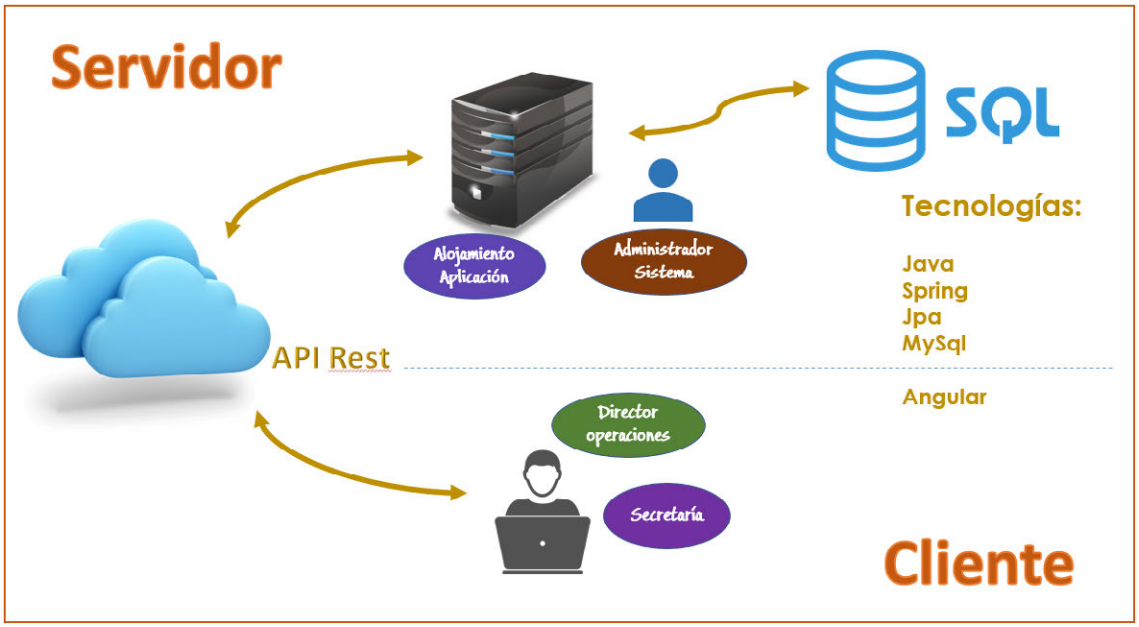


Ilustración 11. Visualizar la solución

6.1.7 ¿Qué nos quita el sueño?

El objetivo de esta actividad es identificar los riesgos en nuestro proyecto para estar alerta ante estos.

Veamos ahora "¿Qué nos quita el sueño?" para nuestra aplicación:

- El desarrollo del Front-end con el framework Angular y su correcto desarrollo debido a la poca experiencia con éste.
- La estrategia DEVOPS a utilizar.
- Autenticación con JWT.
- La correcta sincronización y comunicación entre las distintas tecnologías utilizadas en la integración continua para el desarrollo de la aplicación, ya que si en algún momento falla alguno se verá afectado el correcto funcionamiento de los procesos que están sincronizados.

6.1.8 Calcular el tamaño

El objetivo de esta actividad es intentar calcular el tamaño de nuestro proyecto e indicar si se tardará 1, 3 o 6 meses, no se puede dar un tamaño exacto pero sí un valor aproximado, para eso nos podemos apoyar en el RoadMap, que será la fase posterior a este Inception Deck.

Veamos ahora “Calculo del tamaño” para nuestra aplicación:

Release	Goals	6 semanas	9 semanas
Release 1	Entorno	Crear proyecto	
Release 1	Entorno	Crear entorno DEVOPS	
Release 1	Acceso	Acceso	
Release 2	Gestion empleados		Gestion empleados
Release 2	Gestion clientes		Gestion clientes
Release 3	Gestion envios		Gestion envios
Release 3	Listados		Busquedas

Ilustración 12. Time line

Release	Goals	Feature
Release 1	Entorno	Crear proyecto
Release 1	Entorno	Crear entorno DEVOPS
Release 1	Acceso	Login
Release 2	Gestion empleados	Gestion empleados
Release 3	Gestion clientes	Gestion clientes
Release 3	Gestion envios	Gestion envios
Release 3	Listados	Busquedas

Ilustración 13. Listado features

Basándose en Road Map de la aplicación:

Estas imágenes son del desarrollo del Road Map (desarrollo y explicación de éste en el apartado), en donde se obtiene una estimación basada en los puntos de esfuerzo y tiempos necesarios para desarrollar cada funcionalidad y documentación de la aplicación.

6.1.9 ¿Cuáles son las prioridades?

El objetivo de esta actividad es indicar el nivel de prioridades que se requieren para las distintas fases que definen el desarrollo del producto como por ejemplo:

- Complejidad de características que influyen en el desarrollo del producto.
- Resolución de conflictos como por ejemplo el tener que mantener el presupuesto y por lo tanto se baja la calidad.

De esta manera se podrá tener claro qué se está dispuesto a sacrificar para conseguir los objetivos.

Veamos ahora “¿Cuáles son las prioridades?” para nuestra aplicación:

- Permitir la autenticación con usuario y contraseña a la aplicación.
- Desarrollar la gestión de empleados.

- Desarrollar la gestión de envíos.
- Desarrollar la gestión de clientes.
- Desarrollar las búsqueda.

6.2 Road Map

El roadmap (hoja de ruta) de un proyecto es una vista en perspectiva de alto nivel de todas las entregas, los logros clave y los objetivos en general del proyecto.

Para construir un RoadMap se debe tener claro que éste es un documento vivo y ayuda a responder preguntas clave y estratégicas; permite construir una hoja de ruta y darle a ésta un enfoque de arriba hacia abajo según el nivel de prioridad que queremos dar a cada funcionalidad, no es una lista, sino un árbol o un mapa de ruta a seguir para la elaboración de nuestro proyecto.

Es necesario para poder calcular de manera aproximada los siguientes puntos del Inception Deck:

- Calcular el tamaño
- ¿Cuáles son tus prioridades?

6.2.1 Enfoque multinivel

Es importante tener en cuenta las diferentes visiones del trabajo que se quiere realizar, para esto utilizamos diferentes niveles de visión en el backlog del producto:

1. Goals
2. Features
3. Historias Épicas
4. Historias de usuario (HU)

Al organizarlo de este modo los propietarios del producto y el equipo de desarrollo toman decisiones con mayor facilidad a corto plazo que no afecten negativamente al trabajo.

Veamos cada una de estas visiones del producto.

6.2.1.1 Goals

Representan problemas a resolver. Establecer los objetivos para el lanzamiento de un producto antes de pensar en las características del producto garantiza que los esfuerzos de desarrollo del producto se asignen a sus objetivos respectivos.

Los objetivos de la aplicación son:

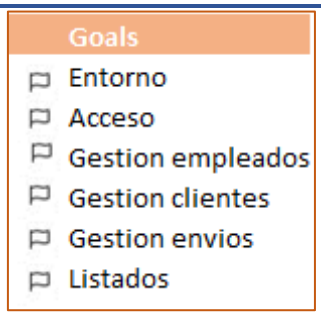


Ilustración 14. Goals de la aplicación

6.2.1.2 Features

Son grandes historias por realizar y representan actividades que llevaran a cabo los usuarios, por lo tanto es una visión de alto nivel del sistema, esta visión está compuesta por varios pasos que se deben desarrollar para obtener la funcionalidad requerida por el usuario. Las características describen las cosas que el sistema tendrá que hacer y los beneficios que el usuario obtendrá, por lo tanto se encuentran al nivel de los requisitos del Software.

Las características de la aplicación son:

Feature	Goals
Crear proyecto	Crear proyecto
Crear entorno DEVOPS	Crear entorno DEVOPS
Login	Acceso
Gestion empleados	Gestion empleados
Gestion clientes	Gestion clientes
Gestion envios	Gestion envios
Busquedas	Busquedas

Ilustración 15.Visión completa de las features del proyecto

6.2.1.2.1 Detalle de las features de la aplicación

Codigo	Feature	Descripción	Goal	Bussines value
F001	**Crear proyecto**	**Generar los proyectos spring y angular proyecto de la aplicación.**	**Crear proyecto2**	**10**
F002	Crear entorno DEVOPS	Crear el entorno necesario para gestionar el control de versiones y la integración continua	Crear entorno DEVOPS	20
F003	Login	Permite acceso a la aplicación a través de usuario y contraseña	Acceso	20
F004	Gestion empleados	Permite gestionar todos los datos relacionados con el alta, baja, modificación y visualización de los empleados de la compañía	Gestion empleados	76
F005	Gestion clientes	Permite gestionar todos los datos relacionados con el alta, baja, modificación y visualización de las empresas contratadoras del servicio de entrega de correo postal	Gestion clientes	76
F006	Gestion envios	Permite gestionar todos los datos relacionados con un sobre o paquete de entrega de correo postal	Gestion envios	80
F007	Busquedas	Realizar listados por diferentes tipo de busqueda	Listados	35

6.2.1.3 Historias de usuarios

Representan pequeñas funcionalidades y describen los requisitos del usuario, se identifican por funcionalidad y no por tareas, indican mediante frases simples lo que tiene que hacer el sistema desde el punto de vista del usuario; mediante las historias de usuario se describe el QUÉ y no el CÓMO.

Una historia de usuario es un elemento con valor de negocio por sí mismo. Junto con las tareas, engloba y forma cada épica (si fuese necesario) y son atómicas, es decir, se pueden poner en producción por sí solas y aportan el valor suficiente a cliente para su utilización.

Las historias se completan a través del documento funcional (o requisitos funcionales) y tienen que contener toda la información posible para que las diferentes áreas puedan trabajar de la manera más adecuada.

Una historia de usuario tiene que cumplir los siguientes criterios:

- Ser atómica, para entrar en un solo Sprint. Aunque puede retrasarse y postergarse, puede llegar el momento en que una historia pueda separarse en dos o en las que sean necesarias.
- Tener valor por sí misma, de cara al cliente, a la hora de finalizar y desplegar en producción.
- Ser estimable. La definición tiene que ser suficiente para que las áreas de trabajo proporcionen una estimación idónea en las reuniones de inicio de sprint.

La historia de usuario tiene que contener de manera obligatoria y lo más específicamente posible una buena definición, unos criterios de aceptación y toda la información adjunta que sea posible.

6.2.1.3.1 Product Backlog:

El produck backlog incluye la lista de historias de usuario e historias técnicas que son necesarias para el desarrollo del software. En el tiempo de desarrollo del proyecto el producto backlog puede cambiar apareciendo nuevas historias de usuario, eliminando alguna o modificando su especificación.

Las Historias de Usuario y técnicas de la aplicación son:

Ilustración 16. Historias de usuario de la aplicación

A continuación se hace la descripción de las historias de usuario y técnicas de la aplicación:

Identificador	HT001	Feature	Crear Proyecto
Nombre	Creación de Proyecto		
Descripción	Como Desarrolladora Quiero crear un proyecto básico Spring para realizar la configuración necesaria de integración y entrega continua; así como también el proyecto básico de Angular para mostrar los datos y consumir los end points		
Valor negocio	10	Prioridad	Muy alta

Identificador	HT002	Feature	Crear entorno DEVOPS
Nombre	Git WorkFlow		
Descripción	Como desarrolladora quiero que se genere el Git workFlow necesario para el control de versiones e integración continua de la aplicación a desarrollar		
Valor negocio	10	Prioridad	Alta

Identificador	HT003	Feature	Crear entorno DEVOPS
Nombre	Workflow GitHub Action test		
Descripción	Como desarrolladora quiero que se hagan las configuraciones necesarias con SonarCloud para las pruebas del software que se va desarrollando en la aplicación		
Valor negocio	10	Prioridad	Alta

Identificador	HU001	Feature	Login
Nombre	Login		
Descripción	Como product owner quiero que se pueda crear a través de spring security por medio de credenciales usuario y contraseña la autenticación de un usuario en la aplicación manejando JWT para el acceso seguro a la aplicación		
Valor negocio	10	Prioridad	Alta

Identificador	HU002	**Feature**	Login
Nombre	Agregar token a peticiones REST		
Descripción	Como desarrolladora quiero agregar el token proporcionado por spring security cuando el login sea correcto en las peticiones REST que desde angular consumen los servicios expuestos por el servidor		
Valor negocio	10	**Prioridad**	Alta

Identificador	HU003	**Feature**	Gestión empleado
Nombre	Análisis funcional de gestión de empleados		
Descripción	Como desarrolladora quiero a través del análisis de gestión de empleados realizar los diagramas de caso de uso, entidad relación y de clases para obtener el documento de requisitos de la gestión de empleados		
Valor negocio	10	**Prioridad**	Alta

Identificador	HU004	**Feature**	Gestión empleado
Nombre	Diseño de gestión de empleados		
Descripción	Como desarrolladora quiero a través de la generación del diagrama de tablas y el diagrama de secuencia obtener el diseño de la gestión de empleados para entender cómo se va a desarrollar esta funcionalidad en la aplicación		
Valor negocio	10	**Prioridad**	Alta

Identificador	HU005	**Feature**	Gestión empleado
Nombre	Alta empleado		
Descripción	Como product owner quiero una opción que permita rellenar un formulario para dar de alta a un empleado		
Valor negocio	20	**Prioridad**	Alta

Identificador	HU006	**Feature**	Gestión empleado

Nombre	Listado empleados		
Descripción	Como product owner quiero una opción en el menú que permita obtener el listado de empleados activos y que se visualice con paginación para poder ver el listado de empleados de la empresa		
Valor negocio	15	**Prioridad**	Alta

Identificador	HU007	**Feature**	Gestión empleado
Nombre	Ver empleado		
Descripción	Como product owner quiero una opción que permita ver la información de un empleado para poder obtener los datos de un empleados en el sistema		
Valor negocio	8	**Prioridad**	Media

Identificador	HU008	**Feature**	Gestión empleado
Nombre	Modificar empleado		
Descripción	Como product owner quiero una opción que permita rellenar un formulario para poder modificar la información de empleados en el sistema		
Valor negocio	8	**Prioridad**	Media

Identificador	HU009	**Feature**	Gestión empleado
Nombre	Eliminar empleado		
Descripción	Como product owner quiero una opción que permita deshabilitar un empleado para no tenerlo en las listas de información de empleados genéricas		
Valor negocio	5	**Prioridad**	Media

Identificador	HU010	**Feature**	Gestión cliente

Nombre	Análisis funcional de gestión de cliente		
Descripción	Como desarrollador quiero a través del análisis funcional de gestión de clientes realizar el diagrama de los casos de uso, diagrama entidad relación y diagrama de clases para obtener el documento de requisitos de la gestión de clientes		
Valor negocio	10	**Prioridad**	Alta

Identificador	HU011	**Feature**	Gestión cliente
Nombre	Diseño de gestión de cliente		
Descripción	Como desarrollador quiero a través de la generación del diagrama de tablas y el diagrama de secuencia obtener el diseño de la gestión de clientes para entender cómo se va a desarrollar esta funcionalidad en la aplicación		
Valor negocio	10	**Prioridad**	Alta

Identificador	HU012	**Feature**	Gestión cliente
Nombre	Alta cliente		
Descripción	Como product owner quiero una opción que permita rellenar un formulario para dar de alta a un cliente		
Valor negocio	20	**Prioridad**	Alta

Identificador	HU013	**Feature**	Gestión cliente
Nombre	Listado cliente		
Descripción	Como product owner quiero una opción en el menú que permita obtener el listado de clientes activos y que se visualice con paginación para poder ver el listado de clientes que tienen un servicio contratado		
Valor negocio	15	**Prioridad**	Alta

Identificador	HU014	**Feature**	Gestión cliente
Nombre	Ver Cliente		

Descripción	Como product owner quiero una opción que permita ver la información de un cliente para poder obtener los datos de un cliente que haya contratado el servicio de entrega postal en el sistema		
Valor negocio	8	**Prioridad**	Media

Identificador	HU015	**Feature**	Gestión cliente
Nombre	Modificar Cliente		
Descripción	Como product owner quiero una opción que permita rellenar un formulario para poder modificar la información de clientes en el sistema		
Valor negocio	8	**Prioridad**	Media

Identificador	HU016	**Feature**	Gestión cliente
Nombre	Eliminar Cliente		
Descripción	Como product owner quiero una opción que permita deshabilitar un cliente para no tenerlo en las listas de información de clientes genéricas		
Valor negocio	5	**Prioridad**	Media

Identificador	HU017	**Feature**	Gestión envío
Nombre	Análisis funcional de gestión guías		
Descripción	Como product owner quiero a través del análisis funcional de gestión de guías realizar los diagramas de casos de uso, diagrama entidad relación y diagrama de clases para obtener el documento de requisitos de la gestión de guías		
Valor negocio	10	**Prioridad**	Alta

Identificador	HU018	**Feature**	Gestión envío

Nombre	Diseño de gestión guía		
Descripción	Como product owner quiero a través de la generación del diagrama de tablas y el diagrama de secuencia obtener el diseño de la gestión de guías para entender cómo se va a desarrollar esta funcionalidad en la aplicación		
Valor negocio	10	**Prioridad**	Alta

Identificador	HU019	**Feature**	Gestión envío
Nombre	Alta Guía		
Descripción	Como product owner quiero una opción que permita rellenar un formulario para dar de alta una guía		
Valor negocio	30	**Prioridad**	Alta

Identificador	HU023	**Feature**	Búsqueda
Nombre	Búsqueda de guía parámetros no fecha		
Descripción	Como Product Owner quiero una opción ene el menú que permita generar las opciones de búsqueda de guía con las opciones: estado de guía y numero de guía para obtener el de listado información sobre los envíos a través de diferentes parámetros de búsqueda		
Valor negocio	20	**Prioridad**	Alta

Identificador	HU020	**Feature**	Gestión envío
Nombre	Ver guía		
Descripción	Como product owner quiero una opción que permita ver la información de una guía para poder obtener los datos de una guía de envío postal en el sistema		
Valor negocio	15	**Prioridad**	Alta

Identificador	HU021	**Feature**	Gestión envío
Nombre	Modificar guía		

Descripción	Como product owner quiero una opción que permita rellenar un formulario para poder modificar la información de una guía en el sistema		
Valor negocio	15	**Prioridad**	Media

Identificador	HU022	**Feature**	Gestión envío
Nombre	Eliminar guía		
Descripción	Como product owner quiero una opción que permita deshabilitar una guía para no tenerla en las listas de información de búsqueda genéricas		
Valor negocio	15	**Prioridad**	Media

Identificador	HU023	**Feature**	Búsqueda
Nombre	Búsqueda de guías por fechas		
Descripción	Como Product Owner quiero que se generen las opciones de búsqueda de guía con las opciones: fecha de entrada, fecha de cargue, fecha entrega para obtener información sobre los envíos a través de diferentes parámetros de búsqueda		
Valor negocio	15	**Prioridad**	Alta

6.2.2 TimeLine

Release	Goals	Feature	Release Start	Release end
Release 1	Entorno	Crear proyecto	semana 1	semana 6
Release 1	Entorno	Crear entorno DEVOPS	semana 1	semana 6
Release 1	Acceso	Login	semana 1	semana 6
Release 2	Gestion empleados	Gestion empleados	semana 7	semana 13
Release 2	Gestion clientes	Gestion clientes	semana 7	semana 13
Release 3	Gestion envios	Gestion envios	semana 14	semana 21
Release 3	Listados	Busquedas	semana 14	semana 21

Ilustración 17. Road Map de la aplicación

7 Estudio teórico para el Desarrollo de la aplicación

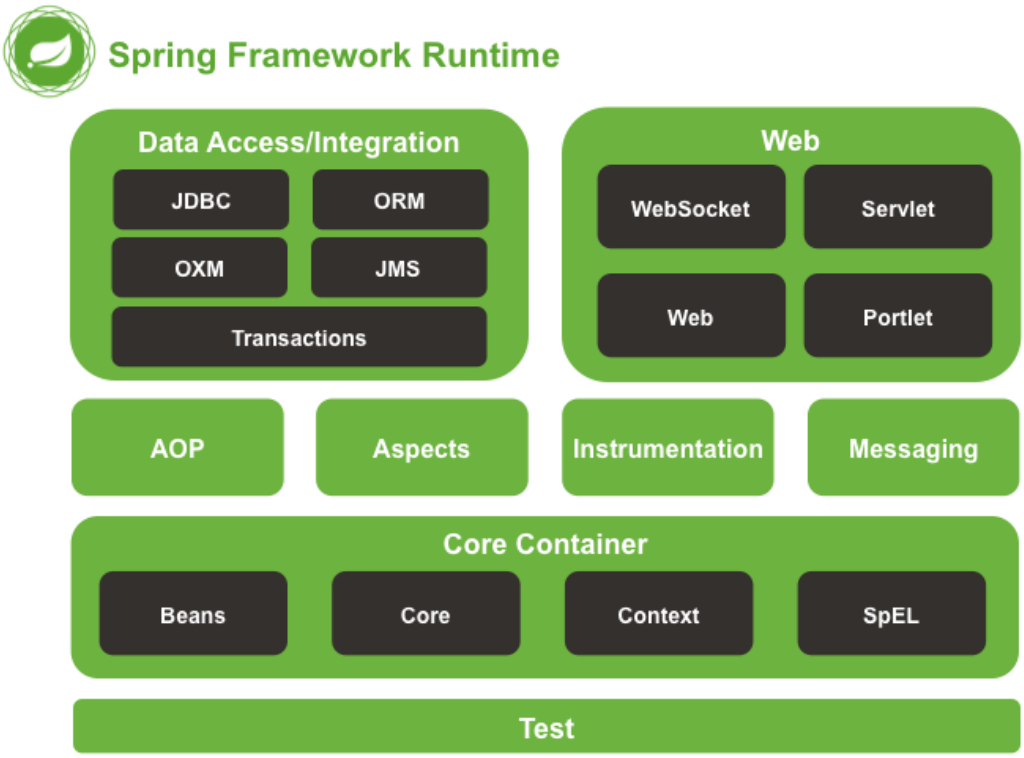
7.1 Tecnologías del entorno de desarrollo

A continuación se describen las tecnologías indicadas en el apartado del inception deck Visualizar la Solución .

7.1.1 Spring framework y Spring Boot

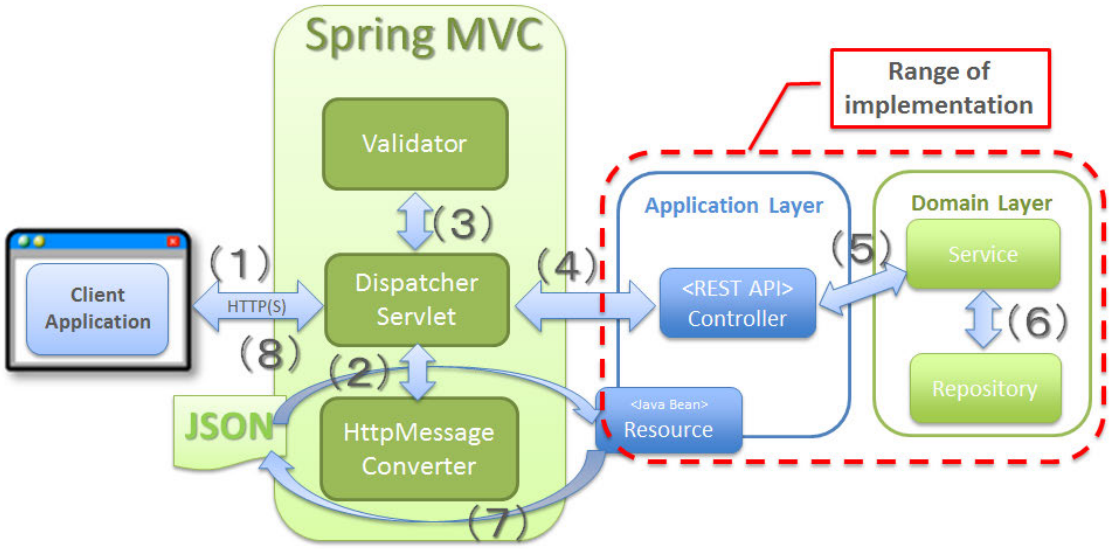
Spring framework:

Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control que gestiona el ciclo de vida de los objetos y como se relacionan entre ellos, es de código abierto para la plataforma JEE. Proporciona una gran infraestructura que permite que el desarrollador se dedique a la lógica de la aplicación, Spring Framework comprende diversos módulos opcionales que proveen un rango de servicios.



- Core Container: La parte fundamental de este framework es el módulo Core, y los adyacentes Bean y Context. Proveen toda la funcionalidad para la inyección de dependencias, permitiéndole administrar la funcionalidad del contenedor de Beans.

- AOP: Se trata de un módulo que permite utilizar el paradigma de Programación Orientada a Aspectos (Aspect Oriented Programming). No se entrará en detalle con respecto a éste porque no se utilizará en el desarrollo de la aplicación.
- Aspects: Este módulo proporciona integración con AspectJ, que es otro potente AOP (Programación Orientada a Aspectos) framework aparte del que ofrece Spring AOP.
- Messaging: Registro configurable de objetos receptores de mensajes, para el consumo transparente desde la a través de JMS (API servicio de mensajes Java), una mejora del envío de mensajes sobre las API JMS estándar.
- Data: Se trata de un gran módulo, formado por múltiples submódulos, y que permite simplificar el acceso y persistencia de datos. Spring Data nos proporciona soporte para usar base de datos relacionales (JDBC), ORMs (como por ejemplo JPA, Hibernate, …) e incluso modelos de persistencia NoSQL (como por ejemplo, MongoDB). En la aplicación a desarrollar se utilizará JPA e Hibernate.
- Test Soporte de clases para desarrollo de unidades de prueba e integración con algunos frameworks de pruebas, tales como JUnit, TestNG, etc.
- Web: Este módulo nos permitirá implementar el patrón Modelo-Vista-Controlador (MVC) de una manera sencilla y limpia, haciendo uso de forma transparente también de otros patrones de diseño, como FrontController. De esta forma, podemos separar limpiamente la lógica de negocio de la presentación de los datos y el acceso a los mismos. Además de aplicaciones que implican el uso de vistas y formularios, también podremos crear servicios web (por ejemplo, al estilo REST) de una forma sencilla y rápida. La aplicación se realizará utilizando este módulo.



Spring Framework es muy potente aunque la configuración inicial y la preparación de las aplicaciones son tareas bastante tediosas, motivo por el cual se utilizara Spring Boot

ya que simplifica el proceso al máximo gracias a sus dos principales mecanismos Contenedor de aplicaciones integrado y Starters.

Spring Boot:

Spring Boot es una herramienta que nos permite crear una aplicación Spring de manera rápida para desplegar rápidamente nuestra aplicación, evitando tener que realizar manualmente ciertas configuraciones siendo sus dos principales mecanismos.

- Contenedor de aplicaciones integrado: permite compilar las aplicaciones Web como un archivo .jar que se puede ejecutar como una aplicación Java normal (como alternativa a un archivo .war que se desplegaría en un servidor de aplicaciones como Tomcat).
  Esto lo consigue integrando el servidor de aplicaciones en el propi**o** .jar y levantándolo cuando se arranca la aplicación. De esta forma, se pueden distribuir las aplicaciones de una forma mucho más sencilla, al poder configurar el servidor junto con la aplicación.

- Starters: proporciona una serie de dependencias, llamadas *starters*, que se pueden añadir al proyecto dependiendo de lo que se necesite: crear un controlador REST, acceder a una base de datos usando JDBC, conectar con una cola de mensajes Apache ActiveMQ, etc.
  Una vez se añade un *starter*, éste proporciona todas las dependencias que se necesitan, tanto de Spring como de terceros. Además, los *starters* vienen configurados con valores por defecto, que pretenden minimizar la necesidad de configuración a la hora de desarrollar.

7.1.2 Angular framework

Angular es una plataforma y un framework para crear aplicaciones de una sola página en el lado del cliente usando HTML y TypeScript. Angular está escrito en TypeScript. Implementa la funcionalidad básica y opcional como un conjunto de bibliotecas TypeScript que importas en tus aplicaciones.

La arquitectura de una aplicación en Angular se basa en ciertos conceptos fundamentales. Los bloques de construcción básicos son los NgModules, que proporcionan un contexto de compilación para los componentes. Los NgModules recopilan código relacionado en conjuntos funcionales; una aplicación de Angular se define por un conjunto de NgModules. Una aplicación siempre tiene al menos un módulo raíz que permite el arranque y generalmente tiene muchos más módulos de funcionalidad.

Los componentes definen vistas, que son conjuntos de elementos de la pantalla que Angular puede elegir y modificar de acuerdo con la lógica y los datos de tu programa.

Los componentes usan servicios, los cuales proporcionan una funcionalidad específica que no está directamente relacionada con las vistas. Los proveedores de servicios pueden inyectarse en componentes como dependencias, haciendo que tu código sea modular, reutilizable y eficiente.

Los módulos, componentes y servicios son clases que usan decoradores. Estos decoradores indican su tipo y proporcionan metadatos que le indican a Angular cómo usarlos.

Los metadatos para una clase componente son asociados con una plantilla que define una vista. Una plantilla combina HTML ordinario con directivas de Angular y enlace markup que permiten a Angular modificar el HTML antes de mostrarlo para su visualización.

Los metadatos para una clase servicio proporcionan la información que Angular necesita para que esté disponible para los componentes a través de la Inyección de Dependencia (ID).

Los componentes de una aplicación suelen definir muchas vistas, ordenadas jerárquicamente. Angular proporciona el servicio Router para ayudarte a definir rutas de navegación entre vistas. El enrutador proporciona capacidades de navegación sofisticadas en el navegador.

7.1.3 Java Persistence API (JPA)

Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java. Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).

El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos (siguiendo el patrón de mapeo objeto-relacional que es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional) y permitir usar objetos regulares (conocidos como POJOs que es una instancia de una clase que no extiende ni implementa nada en especial).

7.2 Integración Continua

7.2.1 Git Hub

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones ***Git*** que es un software de control de versiones.

7.2.1.1 Control versiones

GitHub está pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el trabajo en equipo. Entre ellas, caben destacar:

- Una **wiki** para el mantenimiento de las distintas versiones de las páginas.
- Un **sistema de seguimiento de problemas** que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- Una **herramienta de revisión de código**, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- Un **visor de ramas** donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

7.2.1.2 GitHub Actions

Se utilizará GitHub actions para el control de calidad con SonarQube y las pruebas automatizadas de Junit.

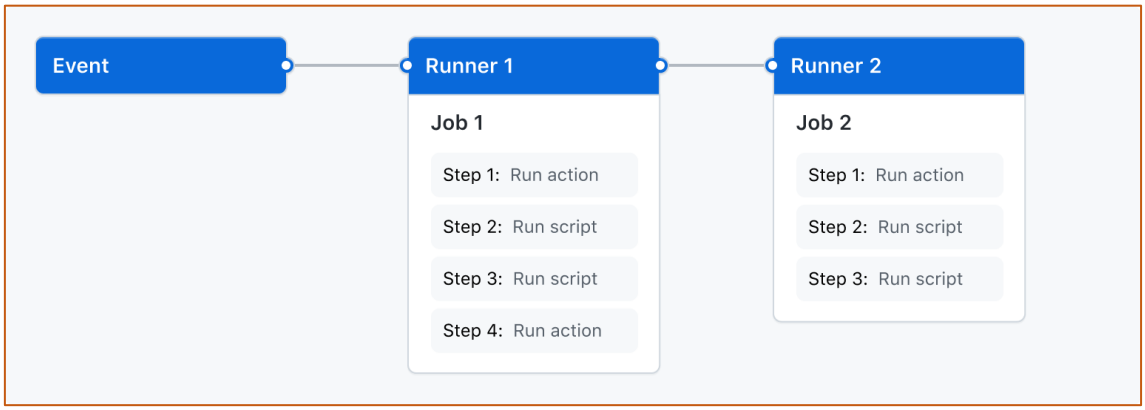
GitHub Actions es una plataforma de integración y despliegue continuos (IC/DC) que permite automatizar el mapa de compilación, pruebas y despliegue. Puede crear flujos de trabajo así como también crear y probar cada solicitud de cambios en un repositorio o desplegar solicitudes de cambios fusionadas a producción.

GitHub Actions va más allá de solo DevOps y permite ejecutar flujos de trabajo cuando otros eventos suceden en un repositorio. Por ejemplo, puede ejecutar un flujo de trabajo para que agregue automáticamente las etiquetas adecuadas cada que alguien cree una propuesta nueva en el repositorio.

GitHub proporciona máquinas virtuales Linux, Windows y macOS para que ejecutes tus flujos de trabajo o puedes hospedar tus propios ejecutores auto-hospedados en tu propio centro de datos o infraestructura en la nube.

Un workFlow es un proceso automatizado configurable que ejecutará uno o más jobs. Los workFlows se definen mediante un archivo de YAML que se verifica en el repositorio y se ejecutará cuando lo active un evento dentro de éste, también puede activarse manualmente o en una programación definida.

Se puede configurar un workFlow de GitHub Actions para que se desencadene cuando se produzca un evento en el repositorio. Un workFlow contiene uno o varios trabajos que se pueden ejecutar en orden secuencial o en paralelo. Cada trabajo se ejecutará dentro de su propio runner de máquina virtual o dentro de un contenedor, y tendrá uno o varios pasos que pueden ejecutar un script que defina o bien una acción, que es una extensión reutilizable que puede simplificar el flujo de trabajo.



8 Desarrollo de la aplicación

En el desarrollo de este proyecto se realizarán 7 entregas, siendo cada una un sprint que aporta un incremento del producto. Cada sprint tendrá una duración de 3 semanas y se realizará el desarrollo en cada sprint teniendo en cuenta la prioridad de cada historia técnica o de usuario.

8.1 Release 1

Features que se desarrollarán en esta release:

Ilustración 18. Features de la release 1

Historias de usuario que se desarrollaran en esta release:

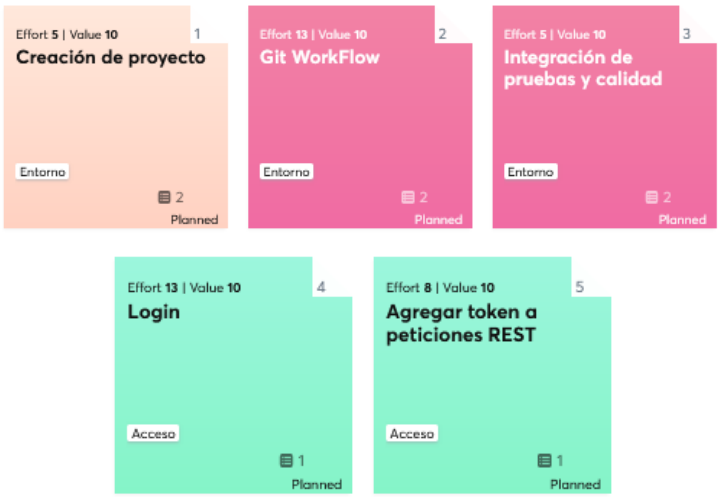


Ilustración 19. Historias de usuario de la release 1

8.1.1 Desarrollo de Sprints Release 1

8.1.1.1 Sprint 1 R1

Sprint planing: 15-05-2023 al 04-06-2023

En este sprint se van a desarrollar las historias de usuario para la configuración del entorno.

A continuación se detallarán las tareas correspondientes a las historias técnicas y de usuario del backlog implicadas en este sprint.

Identificador	S1T1	Historia Técnica	Creación de proyecto
Nombre	Creación de proyecto en parte cliente		

Descripción	Crear un proyecto básico de Angular		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S1T2	**Historia Técnica**	Creación de proyecto
Nombre	Creación de proyecto en parte servidor		
Descripción	Crear un proyecto básico de SpringBoot		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S1T3	**Historia Técnica**	Git WorkFlow
Nombre	Proyecto en aplicación de gestión de proyecto		
Descripción	Preparar el proyecto en aplicación de gestión de proyecto		
Puntos de historia	8	**Prioridad**	Alta

Identificador	S1T4	**Historia Técnica**	Git WorkFlow
Nombre	Creación y enlace de repositorio GitHub proyecto		
Descripción	Crear el repositorio para el proyecto tanto en back-end como en front-end en GitHub y enlazarlo a cada proyecto		
Puntos de historia	5	**Prioridad**	Alta

Identificador	S1T5	**Historia Técnica**	Integración de pruebas y calidad
Nombre	Workflow GitHub Action test		
Descripción	Agregar al proyecto el action test en workflow GitHub		

Puntos de historia	2	**Prioridad**	Alta

Identificador	S1T6	**Historia Técnica**	Integración de pruebas y calidad
Nombre	Workflow GitHub Action Sonar		
Descripción	Agregar al proyecto el action sonar para evaluar la calidad del código en workflow GitHub		
Puntos de historia	3	**Prioridad**	Alta

En este sprint se han creado los proyectos para el servidor y el cliente con la siguiente configuración:

Servidor: Se ha utilizado Spring Boot 2, en el apartado 7: Spring framework y Spring Boot se puede ver la descripción. Se ha utilizado Maven para el control de dependencias entre las cuales se han agregado inicialmente la de Spring security para implementar en sprints futuros el acceso seguro a la aplicación, junit, jpa, lombok y se irán agregando las que se vayan necesitando de ahora en adelante según se vaya avanzando en el desarrollo de la aplicación.

La arquitectura decidida para el proyecto es de tres capas teniendo el siguiente diagrama de paquetes:

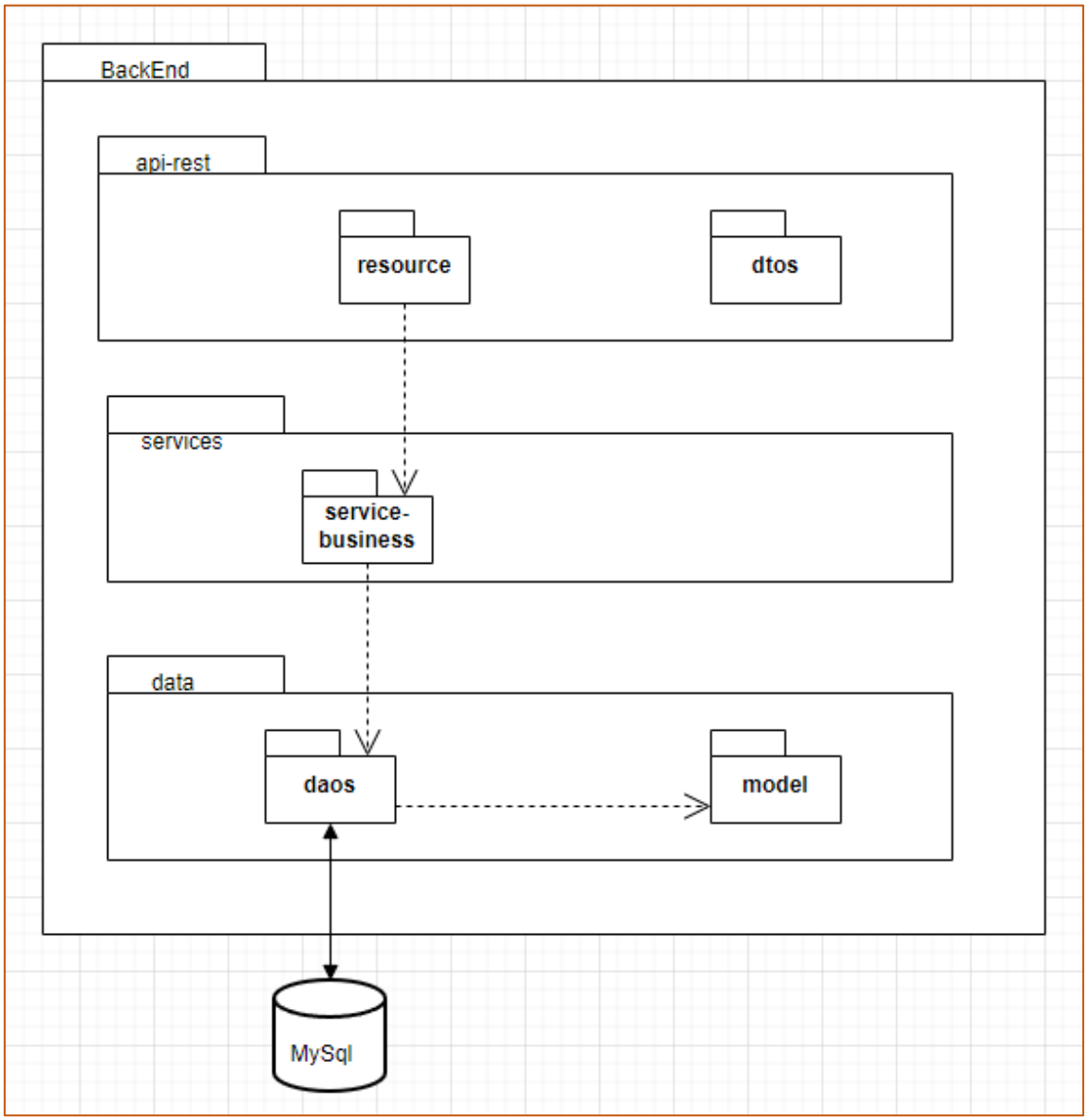


Ilustración 20. Diagrama de paquete backend

Cliente: En la parte cliente se ha creado un proyecto básico de Angular, en el apartado 7: Angular framework se puede ver el detalle de este framework. Se ha agregado Bootstrap para manejar los estilos de las vistas a desarrollar inicialmente.

En la siguiente imagen se puede apreciar el diagrama de paquetes correspondiente a la arquitectura implementada.

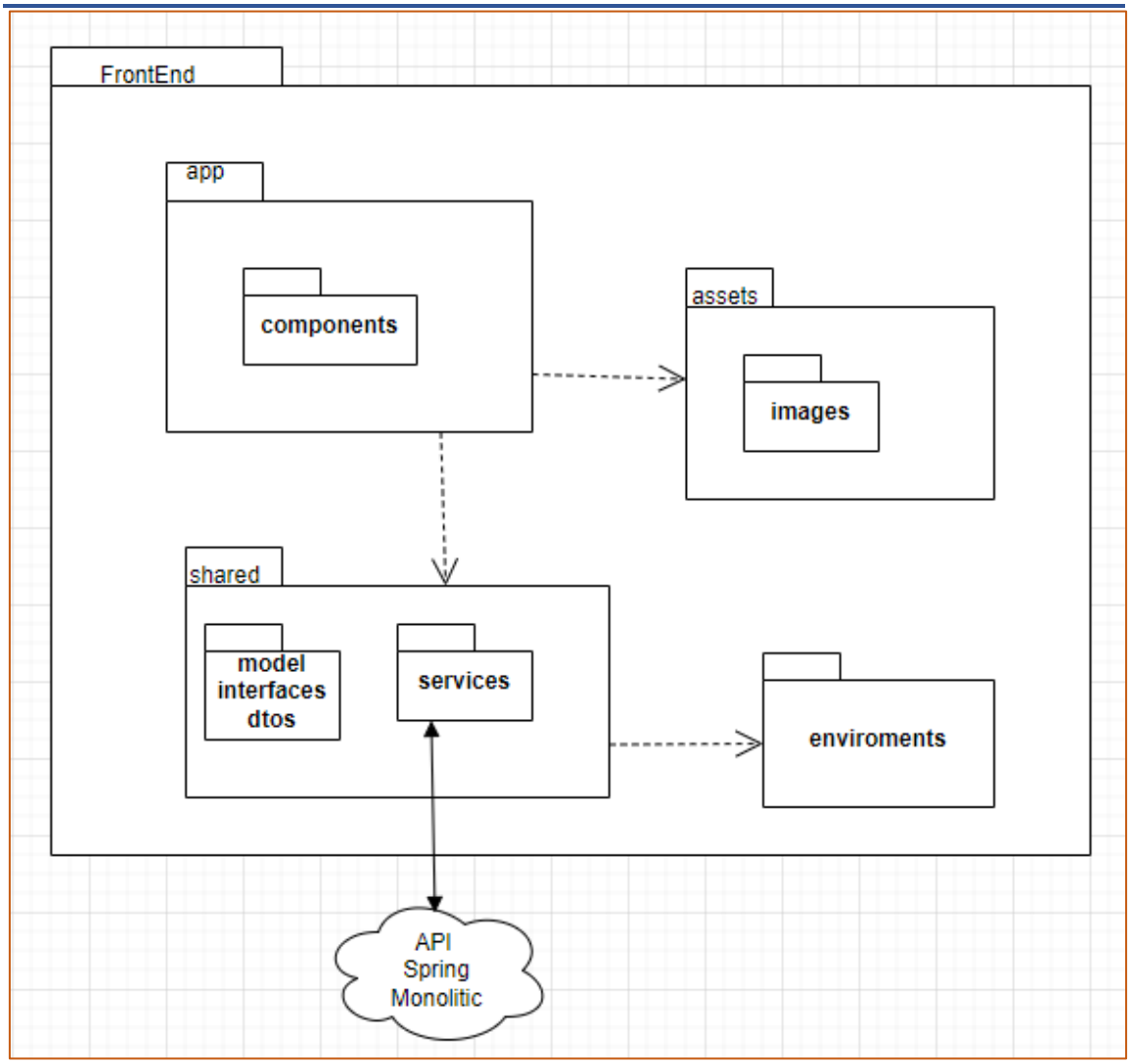


Ilustración 21. Diagrama de paquete frontend

Una vez creados los dos proyectos el siguiente paso ha sido crear el entorno de integración continua a través de un workflow de GitHub cuya definición se ha detallado en el apartado 7: GitHub Actions que soporta la construcción de la aplicación incluyendo la fase de test de Maven. Este procedimiento se ha realizado para la parte backend de la aplicación ya que surgieron problemas en la configuración para la parte frontend y debido al corto plazo de tiempo y para no retrasar el avance se ha decidido dejarlo para el futuro.

En principio se había pensado realizar el despliegue continuo a través de un action de Spring para la parte backend, el cual se ha configurado en el proyecto pero no se ha realizado finalmente el despliegue continuo, una vez la aplicación se construye satisfactoriamente en el action explicado en el párrafo anterior debido a que inicialmente se pensaba desplegar en Heroku pero no ha sido posible debido a que desde noviembre del 2022 ha pasado a ser de pago, con lo cual se deja para el futuro realizar un estudio de plataformas de hosting tanto para el frontend como para el backend y realizar el despliegue a lo que se consideraría el entorno de producción en el hosting elegido.

8.1.1.1.1 Sprint Retrospective

Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no.

En este proyecto al ser individual el alumno tomará todos los roles dentro del equipo.

Sprint Burndown

El propósito del Sprint Burndown es darse cuenta fácilmente y tan pronto como sea posible, si nos encontramos avanzados o retrasados respecto a planificación para poder adaptarnos.

Como se puede apreciar en la siguiente gráfica, el desarrollo de éste se ha realizado en el tiempo estimado aunque en principio se había sido muy optimista y se pensaba que quizá sobraría tiempo, finalmente se ha utilizado todo el tiempo para terminar las tareas planificadas en el sprint.

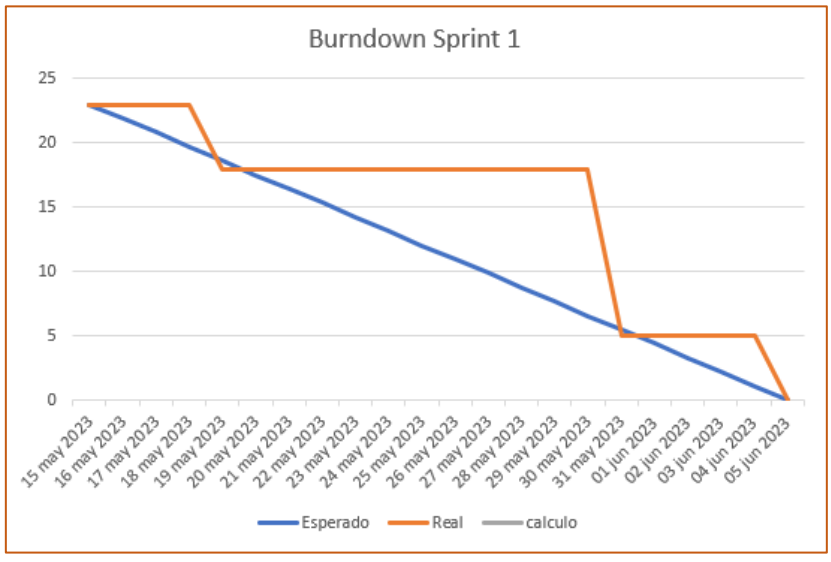


Ilustración 22. Burndown Sprint 1 R1

Para desarrollar el análisis del sprint retrospective desarrollaremos los siguientes apartados en base al desarrollo del Sprint:

- ¿Qué cosas han funcionado bien?
  Finalmente se ha conseguido terminar el Sprint con todas las tareas asignadas a éste aunque se ha tenido que simplificar algunas tareas como la explicada anteriormente que implica el despliegue continuo y la construcción del proyecto angular en GitHub a través de un workflow.
- ¿Cuáles hay que mejorar?
  El tiempo estimado para algunas tareas del Sprint no ha sido correcto ya que han necesitado más esfuerzo del estimado y otras fueron estimadas pensando que necesitaban más esfuerzo, finalmente no lo han necesitado. El problema que ha surgido con respecto a la configuración del entorno de trabajo, por poca practica con el manejo de las diferentes herramientas, ha hecho que al final se haya necesitado más esfuerzo del estimado, este problema ya no afectará más en la planificación puesto que sólo se hace una vez.
- ¿Qué cosas quiere probar hacer en la siguiente iteración?
  Estimar mejor las Historias de usuario y sus tareas.
  Crear una anotación de mejora a futuro para el estudio de hosting de la aplicación y creación de workflow para la parte frontend de la aplicación.
- ¿Qué ha aprendido?
  Se ha aprendido a configurar un entorno de integración continua a través de GitHub con comunicación con SonarCloud.
- Cuáles son los problemas que podrían impedirle progresar adecuadamente.
  La incorrecta estimación de las tareas a realizar puesto que el tiempo es limitado.

8.1.1.2 Sprint 2 R1

Sprint planing: 05-06-2023 al 15-06-2023

En este sprint se van a desarrollar las historias de usuario para el acceso a la aplicación utilizando para ello token JWK en el servidor e Interceptor en la parte cliente.

Historias de usuario a desarrollar en este sprint:

A continuación se detallarán las tareas correspondientes a las historias técnicas y de usuario del backlog implicadas en este sprint.

Identificador	S2T7	Historia de usuario	Login
Nombre	Agregar Login con JWT en el servidor		
Descripción	Crear en el proyecto spring la funcionalidad de login con spring security para que devuelva un token al cliente que se ha autenticado con usuario y contraseña y así éste pueda seguir consumiendo los servicios. También se debe generar la opción de logout para que un usuario pueda cerrar sesión		
Puntos de historia	8	Prioridad	Alta

Identificador	S2T8	Historia de usuario	Login
Nombre	Agregar Login y Home en el cliente		
Descripción	Crear en la parte cliente de la aplicación la vista de login/logout y la lógica necesaria para consumir el servicio de login/logout expuesto en el servidor. Al introducir los datos correctos el usuario accederá a la página de login de la aplicación. Se mostrará la opción contraria al estado de autenticación que tenga el usuario en el menú.		
Puntos de historia	5	Prioridad	Alta

Identificador	S2T9	Historia de usuario	Agregar token a peticiones REST
Nombre	Agregar Token en peticiones REST cliente		
Descripción	Generar lo lógica necesaria con Interceptor de Angular para añadir a las peticiones REST que consumirán los servicios expuestos por el servidor el Token devuelto por el servidor cuando un usuario se autentica en la aplicación		
Puntos de historia	8	Prioridad	Alta

En este sprint se ha realizado el proceso de autenticación de un usuario a través de spring security con JWT y este es el resultado de la generación del código necesario tanto en FrontEnd y BackEnd para que un usuario pueda autenticarse en la aplicación.

Test

Cobertura: En este sprint no se han realizado pruebas unitarias debido a falta de tiempo por los bloqueos y problemas encontrados. Una vez construido el proyecto este es el informe de cobertura de sonar cloud implementado en la tarea: S1T6 de el sprint 1.

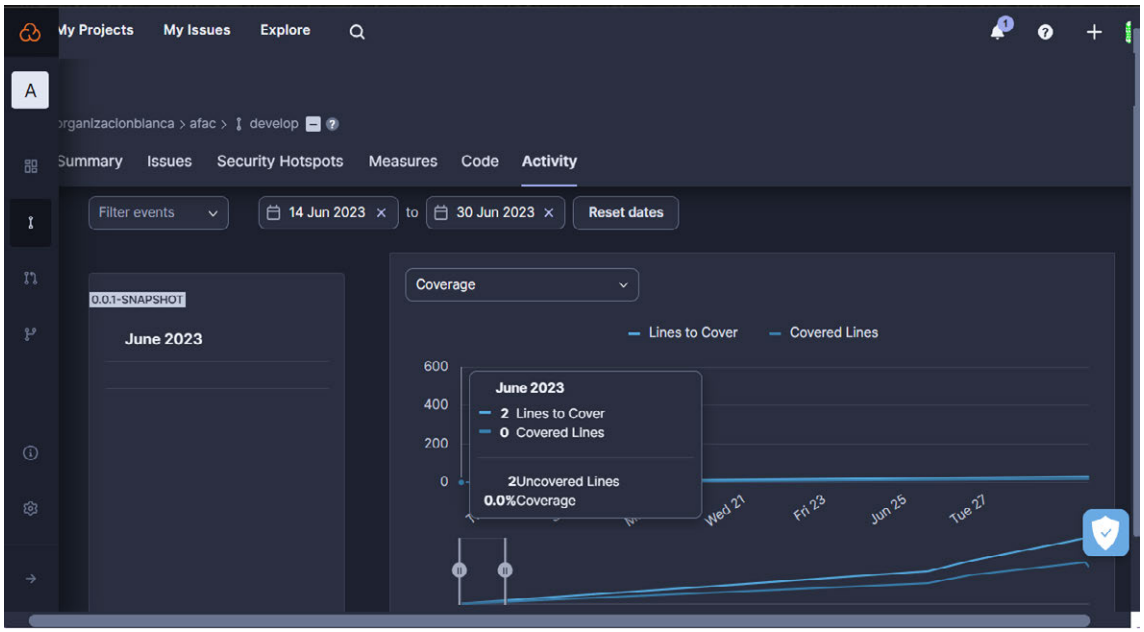


Ilustración 23. Cobertura SonarCloud

Resultado de agregar funcionalidad login/logout

Cuando el usuario introduce correctamente su credenciales se redirige a la página de Home de la aplicación y se pueden ver todas las opciones del menú navbar disponibles en posteriores sprints. Se ha agregado un botón Login/LogOut que estará activo en el sentido contrario del estado de autenticación del usuario y que redirige al login.

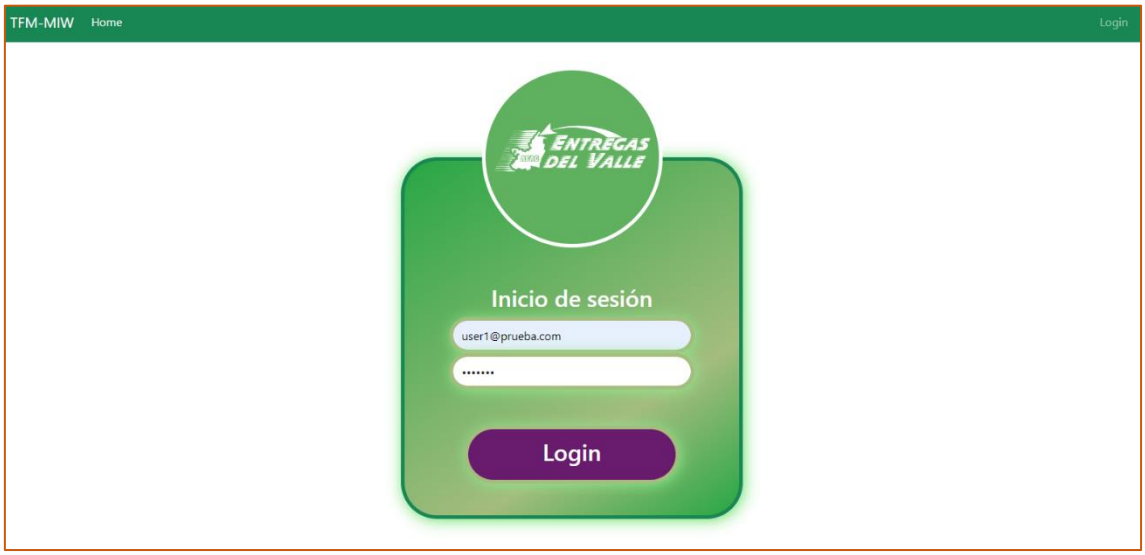


Ilustración 24. Pantalla login

Ilustración 25. Pantalla inicio

8.1.1.2.1 Sprint Retrospective

Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no.

En este proyecto al ser individual el alumno tomará todos los roles dentro del equipo.

Sprint Burndown

A continuación se presenta el gráfico de burdown del sprint. Aunque en la siguiente gráfica se ve que el desarrollo de este sprint se ha realizado en el tiempo, dicho tiempo de manera excepcional se ha tenido que ampliar debido al bloqueo en las dos historias de usuario pues que ha costado más de lo estimado el desarrollo de éstas.

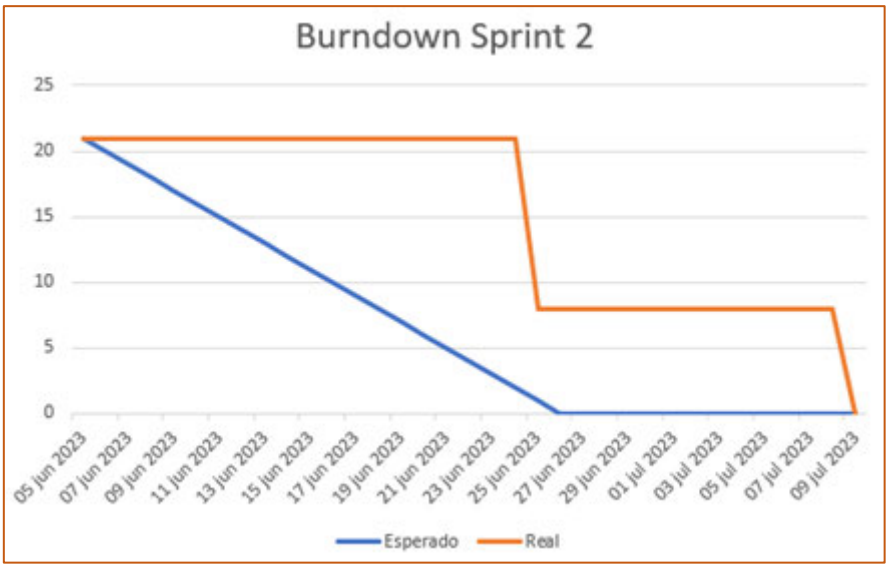


Ilustración 26. Burndown Sprint 2 R1

Para desarrollar el análisis del sprint retrospective desarrollaremos los siguientes apartados en base al desarrollo del Sprint:

- ¿Qué cosas han funcionado bien?
  Se ha conseguido terminar el Sprint con todas las tareas asignadas a éste.
- ¿Cuáles hay que mejorar?

  El tiempo estimado para algunas tareas del Sprint no ha sido correcto ya que han necesitado más esfuerzo y ha habido demasiado estrés para conseguir llegar a la entrega en la fecha estimada. El problema principal que ha surgido ha

sido la generación del proyecto con la nueva versión de spring boot 3 y spring security 5, las cuales han salido justo la semana de creación del proyecto y en éstas se ha cambiado la forma de implementar la autenticación y la creación de JWT, por este motivo se decidió hacer un downgrown a la versión 2 de spring boot. El sprint se ha ampliado de tal forma que ocupa 5 semanas, superando las recomendaciones de duración que es de 2 a 4 semanas cosa que no debería volver a ocurrir ya que si se vuelve a dar está situación es mejor cerrar el sprint y arrastrar de nuevo la historia de usuario al nuevo sprint reestimándola y replanificar el sprint siguiente.

- ¿Qué cosas quiere probar hacer en la siguiente iteración?
  Estimar mejor las Historias de usuario y sus tareas.
- ¿Qué ha aprendido?
  Se ha aprendido a utilizar JWT y spring security así como los Interceptor de Angular.
- Cuáles son los problemas que podrían impedirle progresar adecuadamente.
  La incorrecta estimación de las tareas a realizar puesto que el tiempo es limitado.

8.2 Release 2

Features que se desarrollarán en esta release:

Ilustración 27. Features de la release 2

Historias de usuario que se desarrollaran en esta release:

Ilustración 28. Historias de usuario de la release 2

8.2.1 Desarrollo de Sprints Release 2

8.2.1.1 Sprint 1 R2

Sprint planing: 04-09-2023 al 24-09-2023

En este sprint se van a desarrollar las historias de usuario para el análisis y diseño de la gestión de empleados, así como la visualización del listado de empleados.

A continuación se detallarán las tareas correspondientes a las historias de usuario del backlog implicadas en este sprint.

Identificador	S2T10	**Historia de usuario**	Análisis funcional de gestión de empleados
Nombre	Diagrama de casos de uso de gestión empleado		

Descripción	Generar el diagrama de casos de uso necesario para la gestión de empleados		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S2T11	**Historia de usuario**	Análisis funcional de gestión de empleados
Nombre	Diagrama entidad relación de gestión de empleados		
Descripción	Generar el diagrama de entidad relación necesario para la gestión de empleados		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S2T12	**Historia de usuario**	Análisis funcional de gestión de empleados
Nombre	Diagrama de clases de gestión de empleados		
Descripción	Generar el diagrama de clases necesario para la gestión de empleados		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S2T13	**Historia de usuario**	Diseño de gestión de empleados
Nombre	Paso a tablas gestión empleados		
Descripción	Realizar el diagrama de tablas necesario para la gestión de empleados		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S2T14	**Historia de usuario**	Diseño de gestión de empleados
Nombre	BBDD gestión de empleado		
Descripción	Agregar a la base de datos la parte de estructura de gestión de empleados correspondiente		
Puntos de historia	1	**Prioridad**	Alta

Identificador	S2T15	**Historia de usuario**	Diseño de gestión de empleados
Nombre	Diagrama de secuencia de gestión empleado		
Descripción	Realizar el diagrama de secuencia necesario para la gestión de empleados		
Puntos de historia	5	**Prioridad**	Alta

Identificador	S2T16	**Historia de usuario**	Listado empleados
Nombre	Obtener empleados activos en el Back-end		
Descripción	Crear la lógica necesaria en la parte backend del proyecto para que el servidor pueda exponer un servicio que devuelva una lista paginada de los empleados activos en el sistema.		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S2T17	**Historia de usuario**	Listado empleados
Nombre	Agrega vista empleado en menú		
Descripción	Agregar la opción "Empleado" en el navbar. Al pulsar en esta opción se debe cargar una página que contenga la lista de empleados activos en el sistema. Generar la lógica necesaria para obtener la lista de empleados activos paginada a través del servicio expuesto en el servidor.		

Puntos de historia	3	Prioridad	Alta

8.2.1.1.1 Historia de Usuario Análisis funcional de gestión de empleados

En este apartado se va a desarrollar el análisis funcional de gestión de empleados, el cual ayuda a entender qué se quiere hacer.

8.2.1.1.1.1 Diagrama de casos de uso de gestión empleado

El diagrama de casos de uso representa la forma en como un empleado (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan.

Actor :  un actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Caso de uso:  Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

Relaciones:

- **Asociación:** Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación. Dicha relación se denota con una flecha simple. 
- **Inclusión:** Un caso de uso base incorpora explícitamente el comportamiento de otro en algún lugar de su secuencia. La relación de inclusión sirve para enriquecer un caso de uso con otro y compartir una funcionalidad común entre varios casos de uso, también puede utilizarse para estructurar un caso de uso describiendo sus subfunciones.
  Estas relaciones se representan mediante una flecha discontinua con el estereotipo <<include>>. Algunos casos de uso típicos de inclusión son: comprobar, verificar, buscar, validar, autentificar o login…
- **Extensión:** Un caso de uso base incorpora implícitamente el comportamiento de otro caso de uso en el lugar especificado indirectamente por este otro caso de uso. En el caso de uso base, la extensión se hace en una serie de puntos concretos y previstos en el momento del diseño, llamados **puntos de extensión**, los cuáles no son parte del flujo principal. La relación de extensión sirve para

modelar: la parte opcional del sistema, un sub-flujo que sólo se ejecuta bajo ciertas condiciones o varios flujos que se pueden insertar en un punto determinado. Este tipo de relación produce confusión y no debería utilizarse en exceso. Conviene su uso sólo para insertar un nuevo comportamiento no previsto en un caso de uso existente. Estas relaciones se representan mediante una flecha discontinua con el estereotipo <<extend>>.

- **Generalización:** Un caso de uso (subcaso) hereda el comportamiento y significado de otro, es decir las relaciones de comunicación, inclusión y extensión del super-caso de uso. 

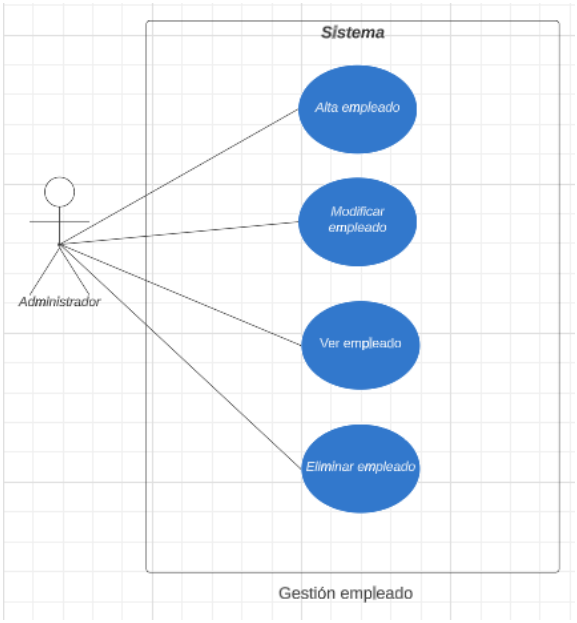


Ilustración 29. diagrama casos de uso gestión empleado

La descripción de los casos de uso se hará a través de la descripción de las tareas en las que se ve involucrada la gestión de empleados.

8.2.1.1.1.2 Diagrama entidad relación de gestión de empleados

Un diagrama o modelo entidad-relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

El modelo entidad-relación se basa en los conceptos descritos a continuación para representar un modelo de la vida real.

- **Entidad:** Una entidad es un objeto que existe y se distingue de otros objetos de acuerdo con sus características llamadas Atributos.
- **Atributos:** Los atributos son las propiedades que describen a cada entidad en un conjunto de entidades.

- **Relación:** Es una asociación o relación matemática entre varias Entidades. Las relaciones también se nombran. Se representan mediante flechas y rombos. Cada entidad interviene en una relación con una determinada cardinalidad (número de instancias o elementos de una entidad que pueden asociarse a un elemento de la otra entidad relacionada). La cardinalidad se representa mediante una pareja de datos, en minúsculas, de la forma (cardinalidad mínima, cardinalidad máxima), asociada a cada una de las entidades que intervienen en la relación. Son posibles las siguientes cardinalidades: (0,1), (1,1), (0,n), (1,n), (m,n). Se informa de las cardinalidades mínimas y máximas con las que intervienen las entidades en la relación.

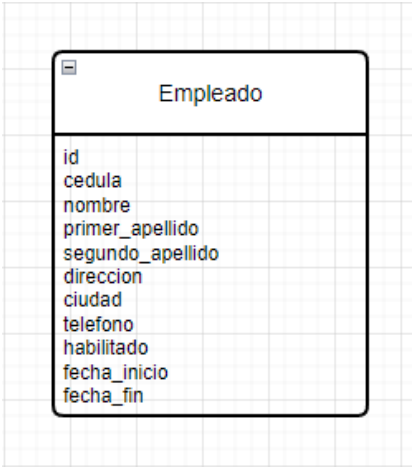


Ilustración 30. Diagrama entidad relación gestión empleados

8.2.1.1.1.3 Diagrama de clases de gestión de empleados

El Diagrama de Clase es el diagrama principal de diseño y análisis para un sistema. Es una estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), relaciones de herencia y las relaciones entre los objetos. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones.

Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

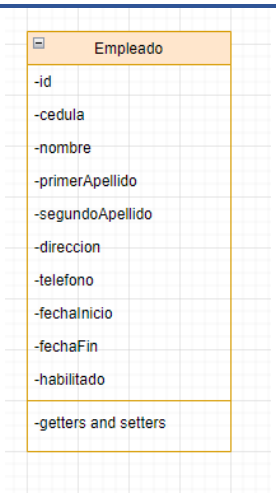


Ilustración 31. Diagrama de clases de gestión de empleados

8.2.1.1.2 Historia de Usuario Diseño de gestión de empleados

En este apartado se va a desarrollar el diseño de la gestión de empleados, el cual ayuda a entender cómo se desarrollará la funcionalidad.

8.2.1.1.2.1 Paso a tablas gestión empleados

El paso a tablas se realiza partiendo del modelo de Entidad Relación, para pasar a tablas todos los datos sin dejarnos nada y que las tablas tengan sentido por si solas tenemos que seguir unos pasos:

- Toda entidad se transforma en una tabla.
- Todo atributo se transforma en una columna dentro de la tabla a la que pertenece.
- El identificador de la entidad se convierte en la clave primaria de la tabla.
- Toda relación N:M se convierte en una tabla que tendrá como clave primaria las dos claves primarias de las entidades que se asocian.
- En las relaciones 1:N la clave primaria de la entidad con cardinalidad 1 pasa a la tabla de la entidad cuya cardinalidad es N.
- En las relaciones 1:1 Si la cardinalidad mínima es:
    - 1:1: Añadir la clave de una tabla cualquiera a la otra tabla + atributos de la relación (si procede)
    - 0:1 o 1:0: Añadir la clave de la tabla “uno” a la tabla “cero” + atributos de la relación (si procede)

  El modelo entidad relación actualmente sólo tiene la información de empleado, así que solo existe la tabla profesor.
  Paso a tablas del modelo Entidad Relación

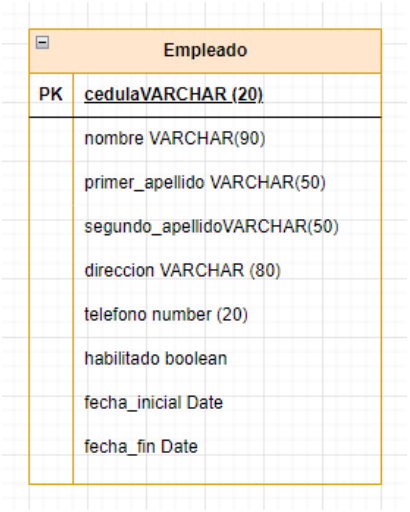


Ilustración 32. Paso a tablas gestión empleado

8.2.1.1.2.1 Script BBDD gestión de empleado

```
CREATE SCHEMA IF NOT EXISTS AFAC DEFAULT CHARACTER SET latin1;

USE AFAC;

DROP TABLE IF EXISTS `EMPLEADO`;

CREATE TABLE `EMPLEADO` (

  `id` bigint(20) NOT NULL AUTO_INCREMENT,

  `cedula` bigint(20) NOT NULL,

  `nombre` varchar(255) NOT NULL,

  `primer_apellido` varchar(255) NOT NULL,

  `segundo_apellido` varchar(255) NOT NULL,

  `direccion` int(11) NOT NULL,

  `telefono` varchar(255) NOT NULL,

  `habilitado` varchar(255) NOT NULL,

  `fecha_inicio` DATE DEFAULT NULL,

  `fecha_fin` iDATE

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1
```

8.2.1.1.2.2 Diagrama de secuencia de gestión empleado

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. A menudo es útil para complementar a un diagrama de clases, pues el diagrama de secuencia se podría describir de manera informal como "el diagrama de clases en movimiento", por lo que ambos deben estar relacionados entre sí (mismas clases, métodos, atributos...).

Existen dos tipos de mensajes: síncronos y asíncronos. Los mensajes síncronos se corresponden con llamadas a métodos del objeto que recibe el mensaje. El objeto que envía el mensaje queda bloqueado hasta que termina la llamada. Este tipo de mensajes se representan con flechas con la punta rellena. Los mensajes asíncronos terminan inmediatamente, y crean un nuevo hilo de ejecución dentro de la secuencia. Se representan con flechas con la punta hueca.

También se representa la respuesta a un mensaje con una flecha discontinua.

Usos: Pueden ser usados de dos formas diferentes:

- De instancia: describe un escenario específico (un escenario es una instancia de la ejecución de un caso de uso).
- Genérico: describe la interacción para un caso de uso. Utiliza ramificaciones ("branches"), condiciones y bucles.

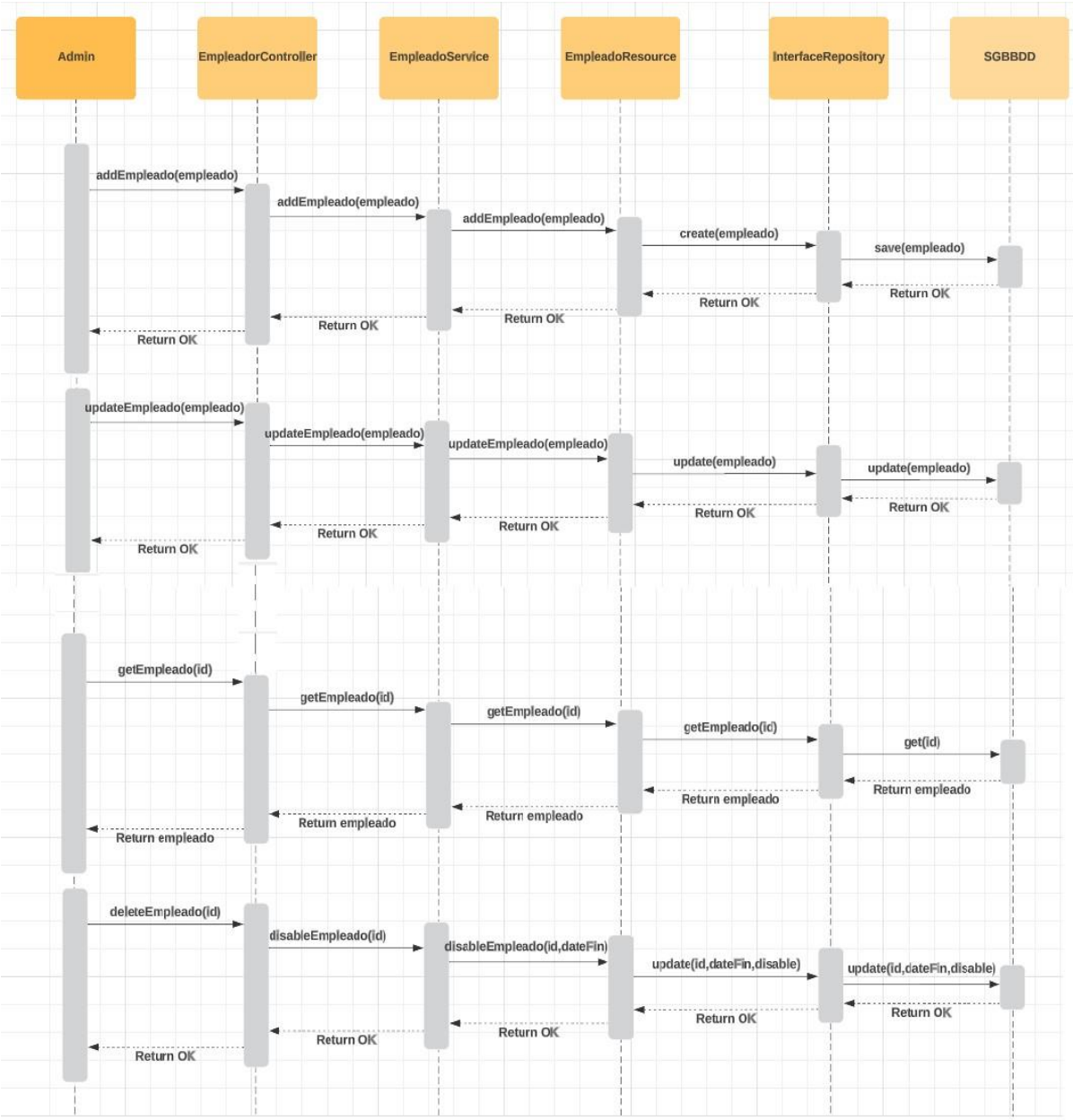


Ilustración 33. Diagrama de secuencia gestión empleados

8.2.1.1.3 Historia de Usuario Listado Empleados

8.2.1.1.3.1 Tarea Obtener empleados activos en el Back-end

Endpoint creados en el backend para la obtención de empleados activos con paginación.

GET/employees/readallactivate:

Permite obtener la lista de empleados activos en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición. Los parámetros para enviar corresponden con:

- Page: la página de inicio de carga
- El tamaño de registros por pagina
- El atributo por el que se quiere realizar la ordenación de los registros
- El orden de los registros que se van a visualizar

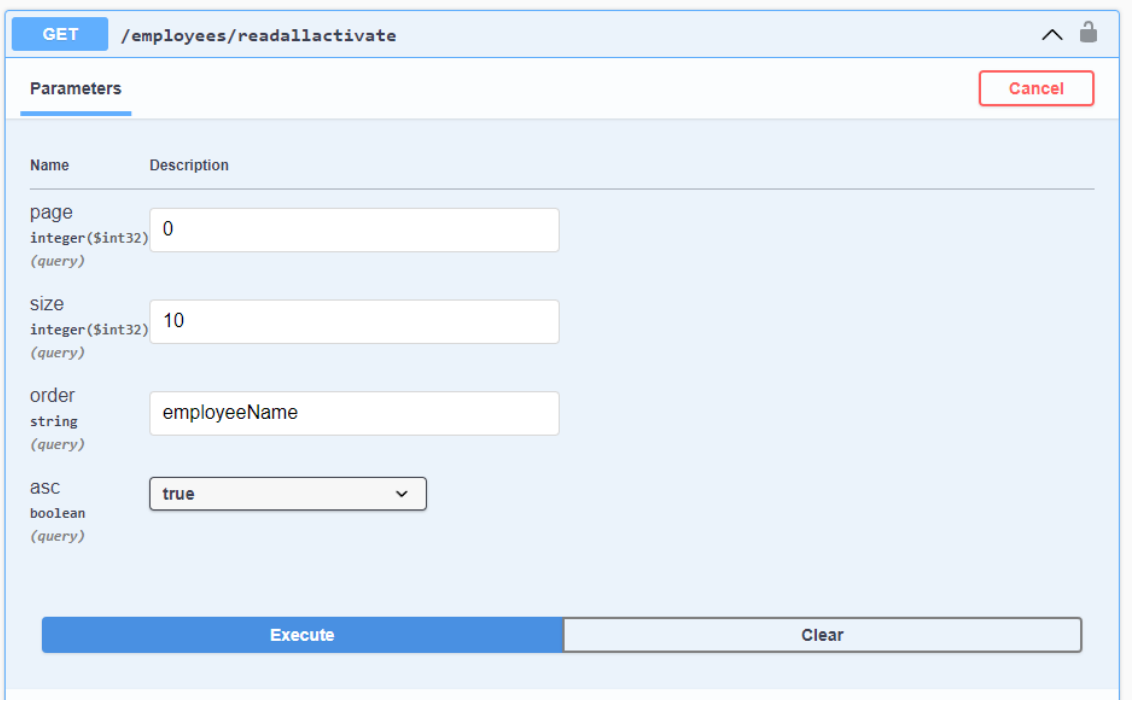


Ilustración 34. EndPoint Get/employees/readallactivate

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para obtener el listado de los empleados activos.

Al pulsar en la opción “Empleado” del navbar, una vez se ha autenticado el usuario en el sistema, se puede ver el listado en una tabla paginada con 10 elementos.

En la siguiente imagen se puede ver el resultado del listado.

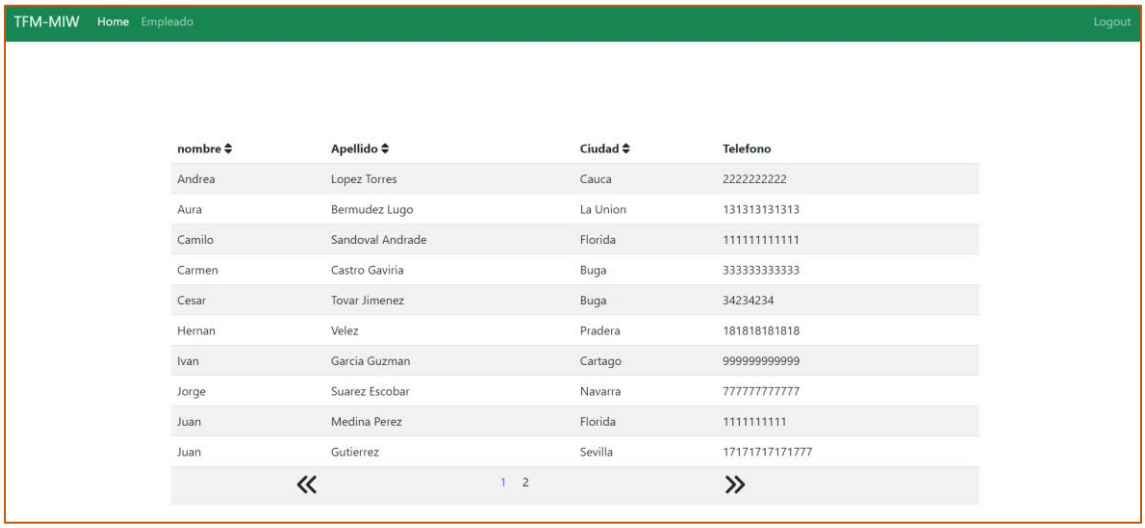
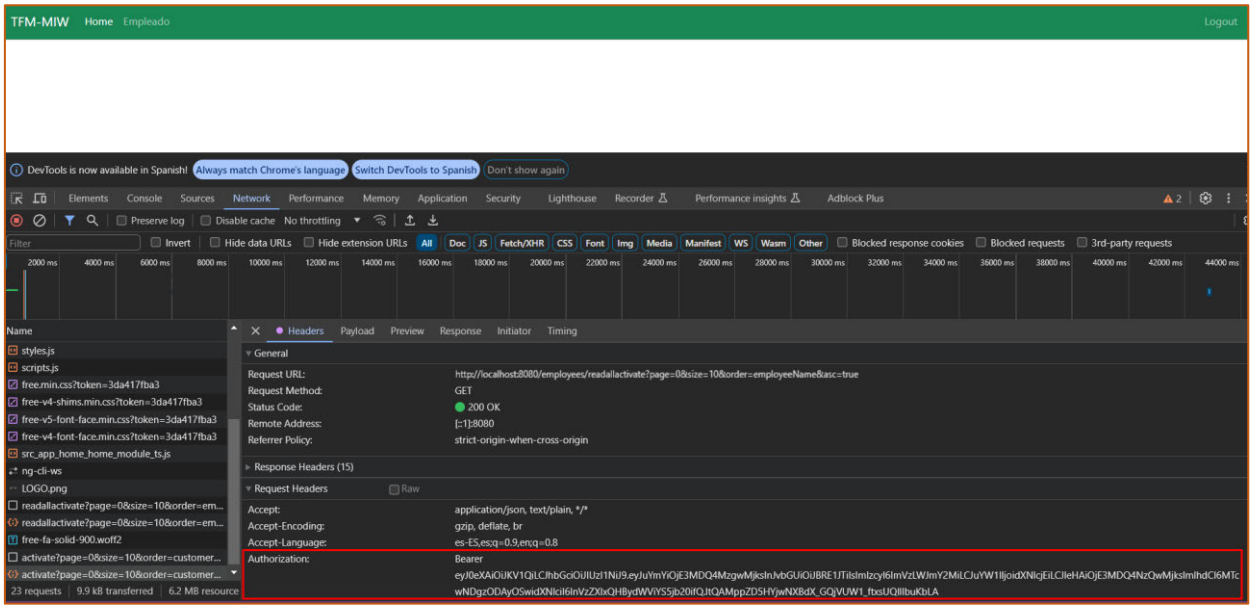


Ilustración 35. Pantalla listado de empleados

Resultado de agregar Token en peticiones REST cliente tarea S2T9 del sprint1

Este es el resultado de la generación del código necesario en FrontEnd para agregar en cada petición REST de consumo de servicio al servidor una vez autenticado el usuario el Bearer token autorization con el token devuelto por el servidor en la autenticación cuando ésta es correcta.



8.2.1.1.4 Sprint Retrospective

Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no.

En este proyecto al ser individual el alumno tomará todos los roles dentro del equipo.

Sprint Burndown

A continuación se presenta el gráfico de burdown del sprint. Como se puede apreciar en la siguiente gráfica, el desarrollo de este sprint se ha realizado en el tiempo estimado ya que se ha aprendido de sprints anteriores para estimar con más realidad las tareas planificadas en el sprint.

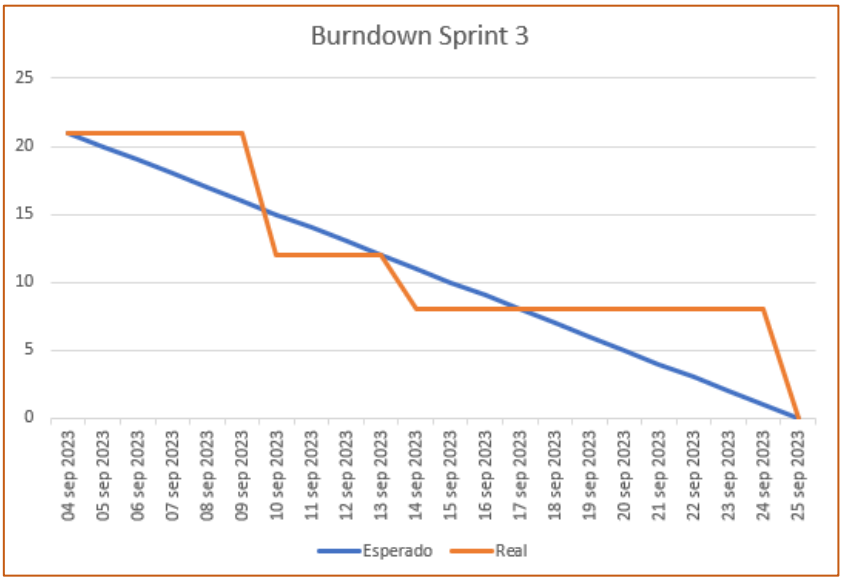


Ilustración 36. Burndown Sprint 1 R2

Para desarrollar el análisis del sprint retrospective desarrollaremos los siguientes apartados en base al desarrollo del Sprint:

- ¿Qué cosas han funcionado bien?
  Finalmente se ha conseguido terminar el Sprint con todas las tareas asignadas a éste.
- ¿Cuáles hay que mejorar?
  El tiempo estimado para algunas tareas del Sprint no ha sido correcto ya que han necesitado más esfuerzo del estimado mientras que otras fueron estimadas pensando que necesitaban más esfuerzo y finalmente no lo han necesitado. El

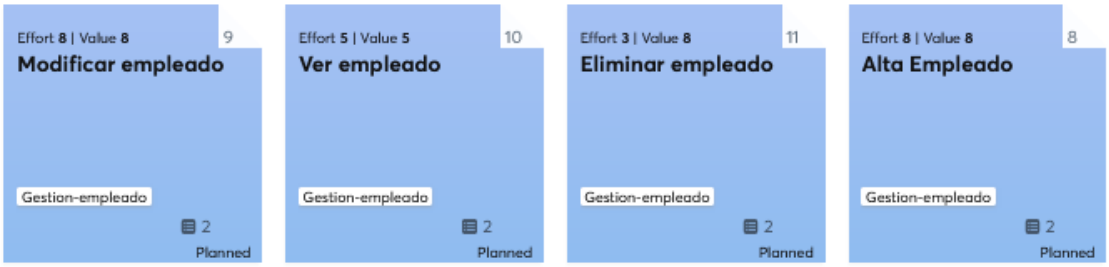
problema que ha surgido ha sido debido al diagrama de secuencia que ha tomado más tiempo de lo pensado.

- ¿Qué cosas quiere probar hacer en la siguiente iteración?
  Mejorar la estimación de las Historias de usuario y sus tareas.
- ¿Qué ha aprendido?
  Se han reforzado los conocimientos sobre análisis y diseño de una funcionalidad.
- Cuáles son los problemas que podrían impedirle progresar adecuadamente.
  La incorrecta estimación de las tareas a realizar puesto que el tiempo es limitado.

8.2.1.2 Sprint 2 R2

Sprint planing: 25-09-2023 al 15-10-2023

En este sprint se van a desarrollar las historias de usuarios de gestión de empleados.



A continuación se detallarán las tareas correspondientes a las historias de usuario del backlog implicadas en este sprint.

Identificador	S4T18	**Historia de usuario**	Alta empleado
Nombre	Crear empleado en servidor		
Descripción	Agregar la lógica necesaria para crear empleado en Spring y exponer el servicio que consuma la parte cliente para esta funcionalidad		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S4T19	**Historia de usuario**	Alta empleado
Nombre	Agregar la opción crear empleado en la parte cliente		
Descripción	Agregar a la vista de listado de empleados la opción de crear empleado, siendo esta un botón en la parte superior izquierda. Agregar la lógica necesaria para la validación del formulario y el envío a través de la api al servicio correspondiente expuesto en el servidor		
Puntos de historia	5	**Prioridad**	Alta

Identificador	S4T20	**Historia de usuario**	Ver empleado
Nombre	Crear obtener empleado en la parte servidor		
Descripción	Agregar la lógica necesaria para obtener la información de un empleado buscando por el id en Spring y exponer el servicio que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S4T21	**Historia de usuario**	Ver empleado
Nombre	Agregar la opción ver empleado en la parte cliente		
Descripción	En Angular, agregar la opción de ver detalles de un empleado. El botón “Ver” debe ir en la tabla de listado de empleados a la derecha de cada empleado listado. Crear la lógica necesaria para la validación del formulario y su envío a través de la api al servicio correspondiente expuesto en el servidor		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S4T22	**Historia de usuario**	Modificar empleado
Nombre	Crear modificar empleado en servidor		

Descripción	Agregar la lógica necesaria para modificar empleado en Spring y exponer el servicio que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	3	**Prioridad**	Medio

Identificador	S4T23	**Historia de usuario**	Modificar empleado
Nombre	Agregar la opción modificar empleado en la parte cliente		
Descripción	En Angular, agregar la opción de modificar empleado. El botón “Modificar” debe ir en la tabla de listados de empleados a la derecha de cada empleado listado. Agregar la lógica necesaria para la validación del formulario y su envío a través de la api al servicio correspondiente expuesto en el servidor		
Puntos de historia	5	**Prioridad**	Medio

Identificador	S4T24	**Historia de usuario**	Eliminar empleado
Nombre	Crear deshabilitar empleado en servidor		
Descripción	Agregar la lógica necesaria para realizar el borrado lógico de un empleado en Spring y exponer el servicio que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	1	**Prioridad**	Baja

Identificador	S4T25	**Historia de usuario**	Eliminar empleado
Nombre	Agregar opción eliminar empleado en la parte cliente		
Descripción	En Angular, agregar la opción de eliminar un empleado. El botón “Eliminar” debe ir en la tabla de listados de empleados a la derecha de cada empleado listado. Agregar lo lógica necesaria para que a		

	través de una ventana de confirmación se consuma el servicio correspondiente expuesto en el servidor		
Puntos de historia	2	**Prioridad**	Baja

8.2.1.2.1 Historia de Usuario Alta Empleado

Endpoint creados en el backend para la creación de empleados.

POST/employees:

Permite la creación de un empleado en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

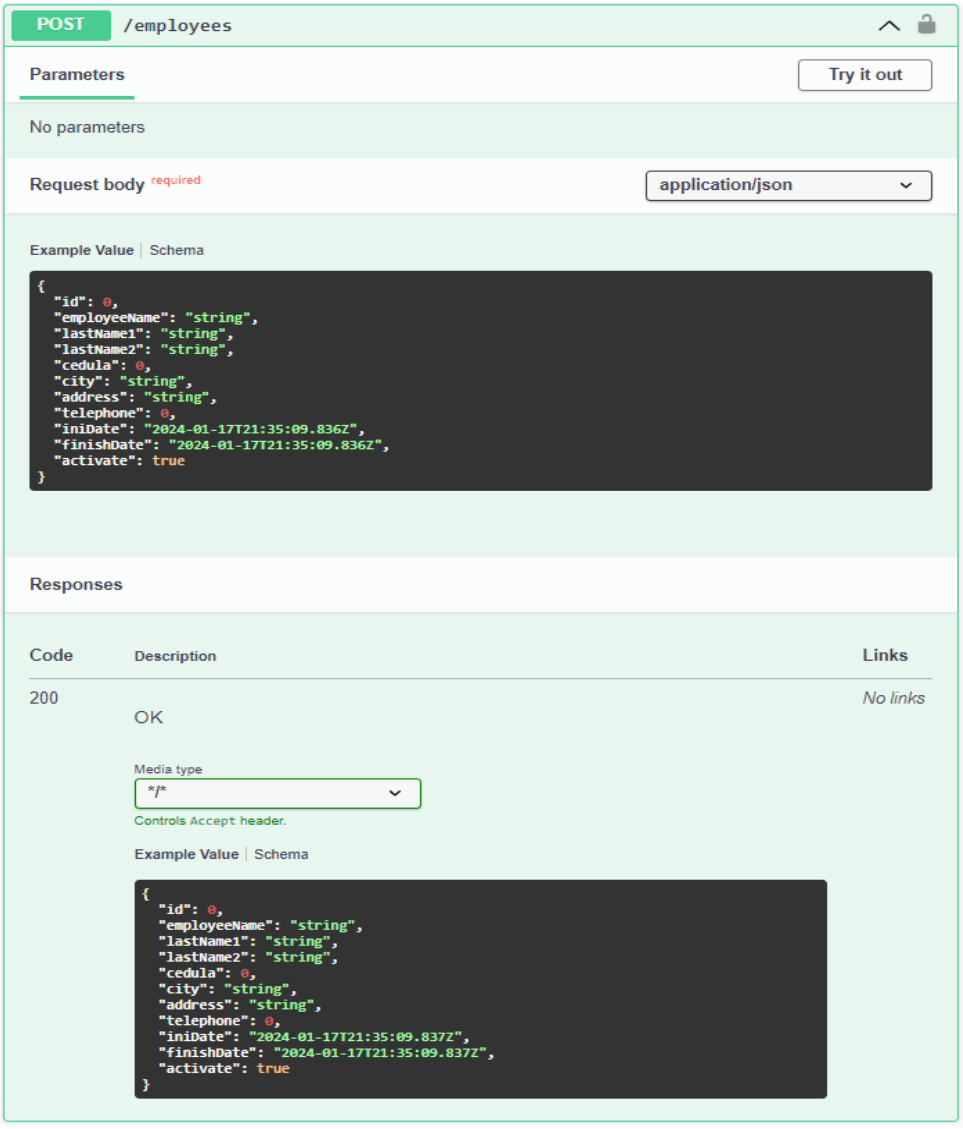


Ilustración 37. POST/employees

GET/ciudad:

Devuelve la lista de ciudades almacenadas en un enumerado en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

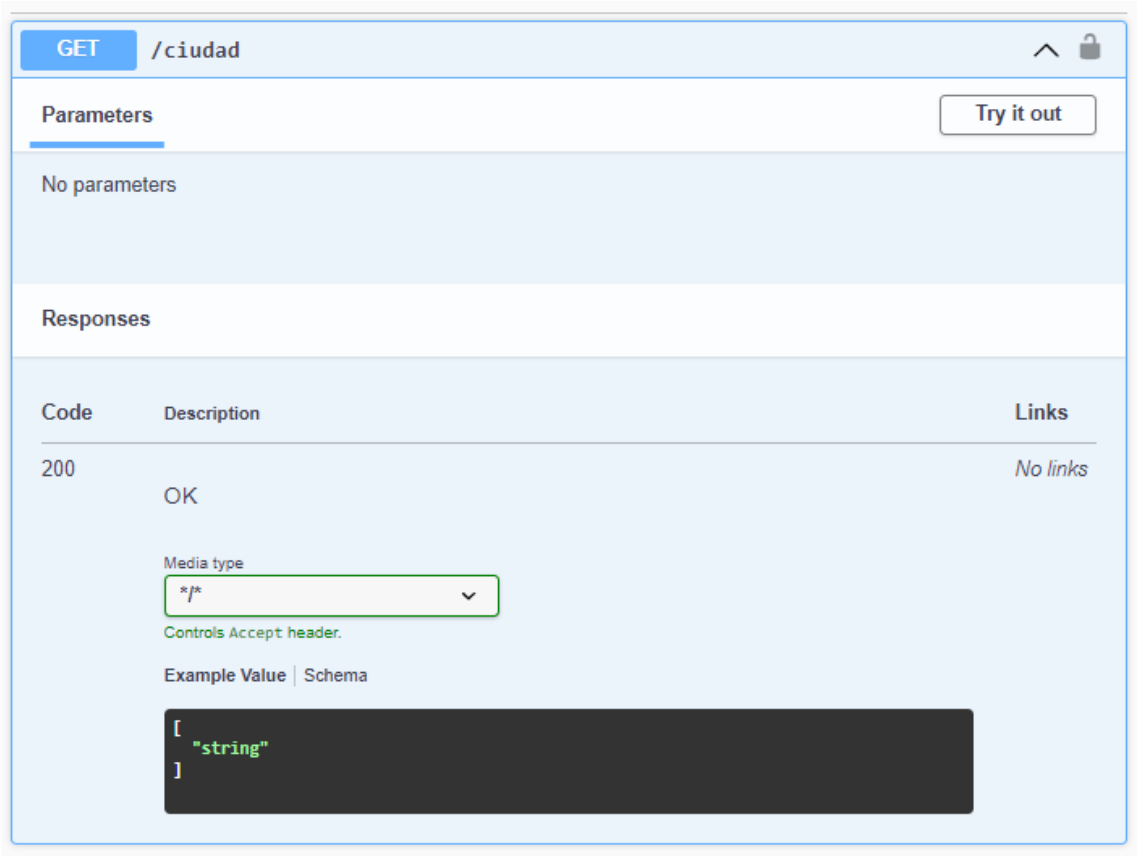


Ilustración 38. GET/ciudad

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para dar de alta a un empleado.

Al pulsar en la opción Empleado del navbar, una vez se ha autenticado el usuario, éste tiene la opción de crear un nuevo empleado.

El formulario para rellenar tiene un campo llamado ciudad que corresponde con la ciudad que tiene asignada un empleado para realizar los repartos de envíos postales. El valor se elige desde un select option que carga el listado de ciudades después de hacer la solicitud de obtener ciudades al servidor.

En la siguiente imagen vemos el formulario que se le proporciona al usuario al pulsar en el botón “Crear Empleado” para introducir los datos de éste. El botón “Enviar” no estará habilitado para ser pulsado hasta que no se hayan introducido todos los datos que son obligatorios en el formulario y tampoco se habilitará si no se cumplen los requisitos de cada campo, en el caso de que no se cumplan los requisitos de cada campo se mostrará un mensaje al usuario indicándolo. También se ha agregado un botón para poner todos los campos del formulario en blanco si el usuario lo desea.

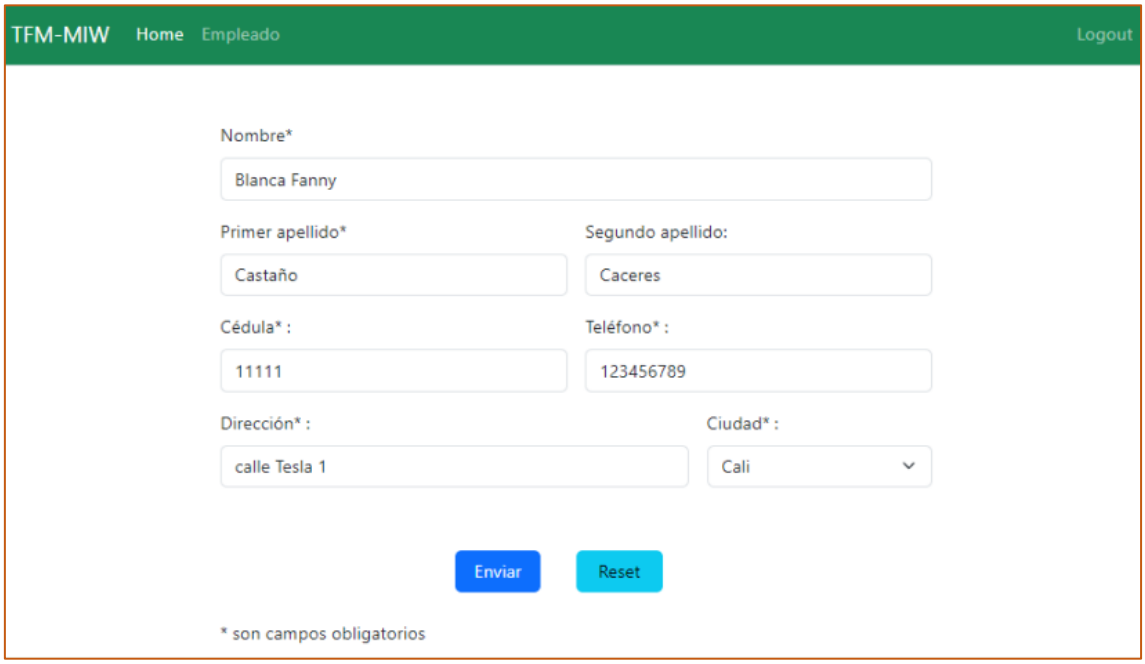


Ilustración 39. Pantalla formulario alta empleado

8.2.1.2.2 Historia de Usuario Ver empleado

Endpoint creado en el backend para la realizar la búsqueda de un empleado por su id.

GET/employees/{id}:

Permite obtener la información de un empleado realizando la búsqueda por su id en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador del empleado y se debe enviar de forma obligatoria.

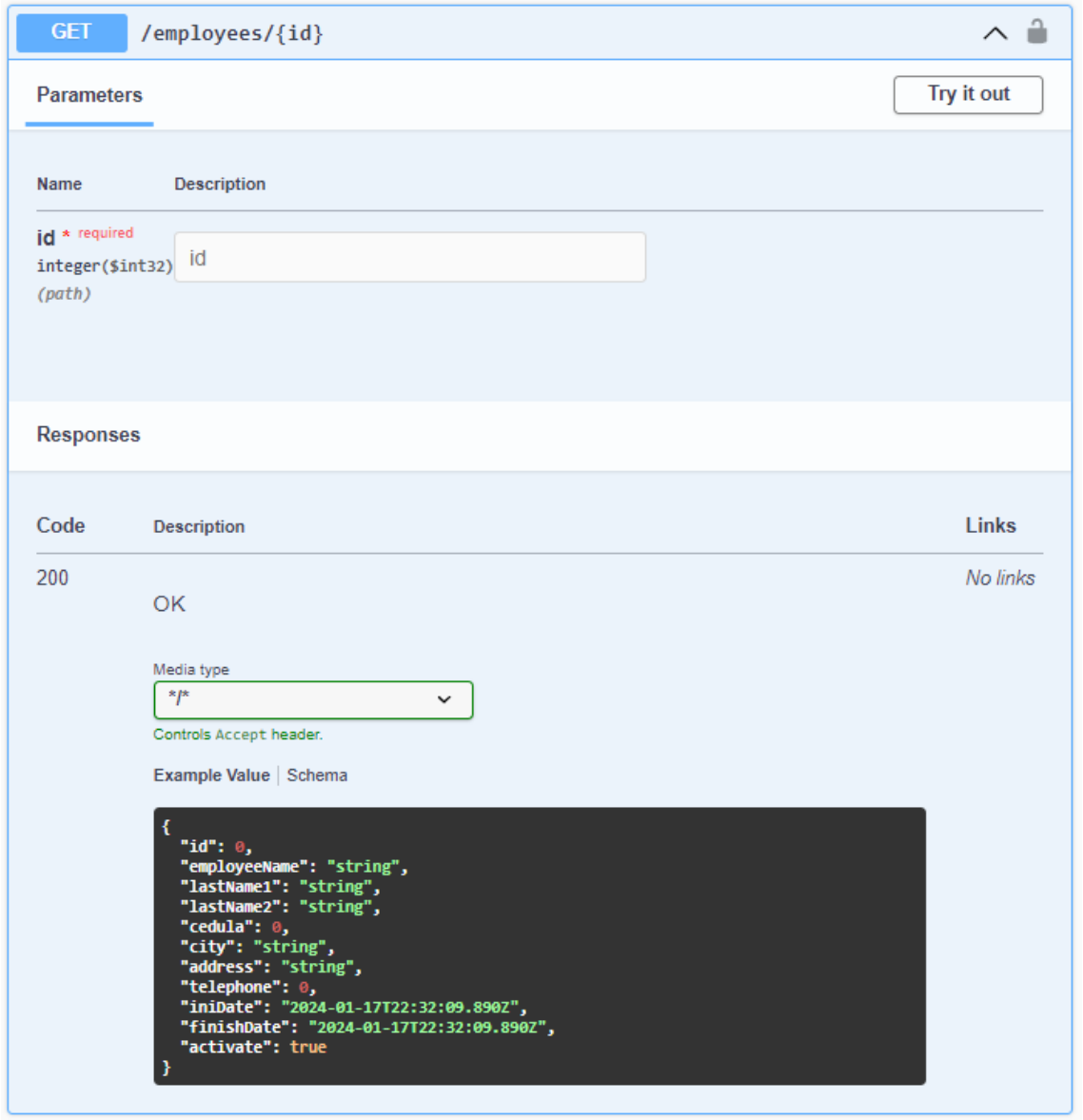


Ilustración 40. GET/employees/{id}

Este es el resultado de la generación del código necesario tanto en FrontEnd y BackEnd para ver los datos detallados de un empleado.

Al pulsar en la opción Empleado del navbar, una vez se ha autenticado el usuario, se carga la lista de empleados en la cual tiene la opción de ver los datos detallados de cada empleado pulsando el botón “Ver” que se encuentra a la derecha de cada empleado de la lista. Se carga la vista detallada de la información de cada empleado en un formulario que muestra la información deshabilitada y contiene un botón de volver que lleva al usuario al listado de empleados.

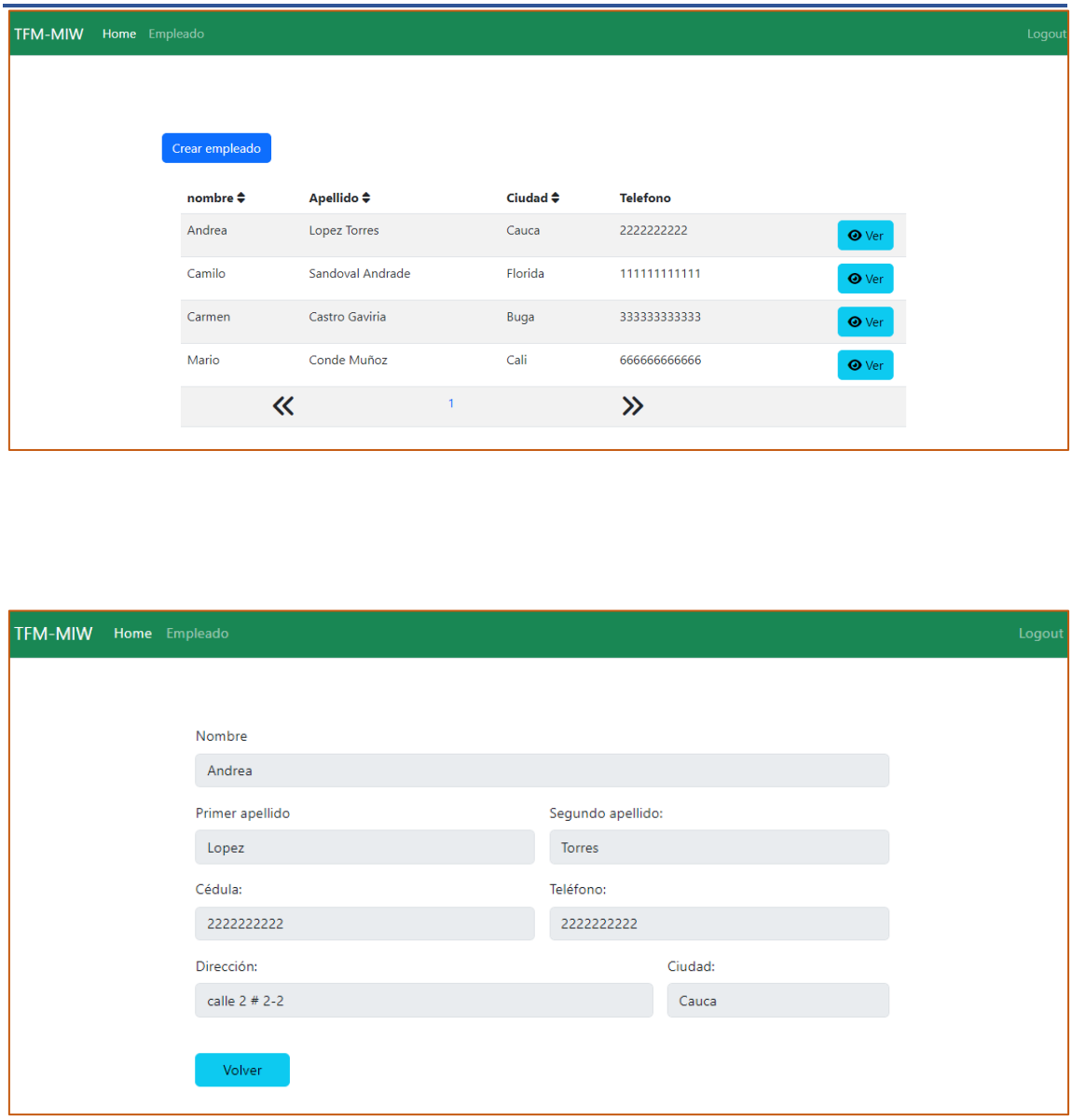


Ilustración 41. Pantalla ver empleado

8.2.1.2.3 Historia de Usuario Modificar empleado

PUT/employees/{id}:

Permite modificar la información de un empleado en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador del empleado y se debe enviar de forma obligatoria para verificar que dicho empleado existe en el sistema antes realizar el proceso de modificación.

La información del objeto enviado en el body corresponde a la información del empleado con los datos que se quieren modificar siendo obligatorio el envío de éste.

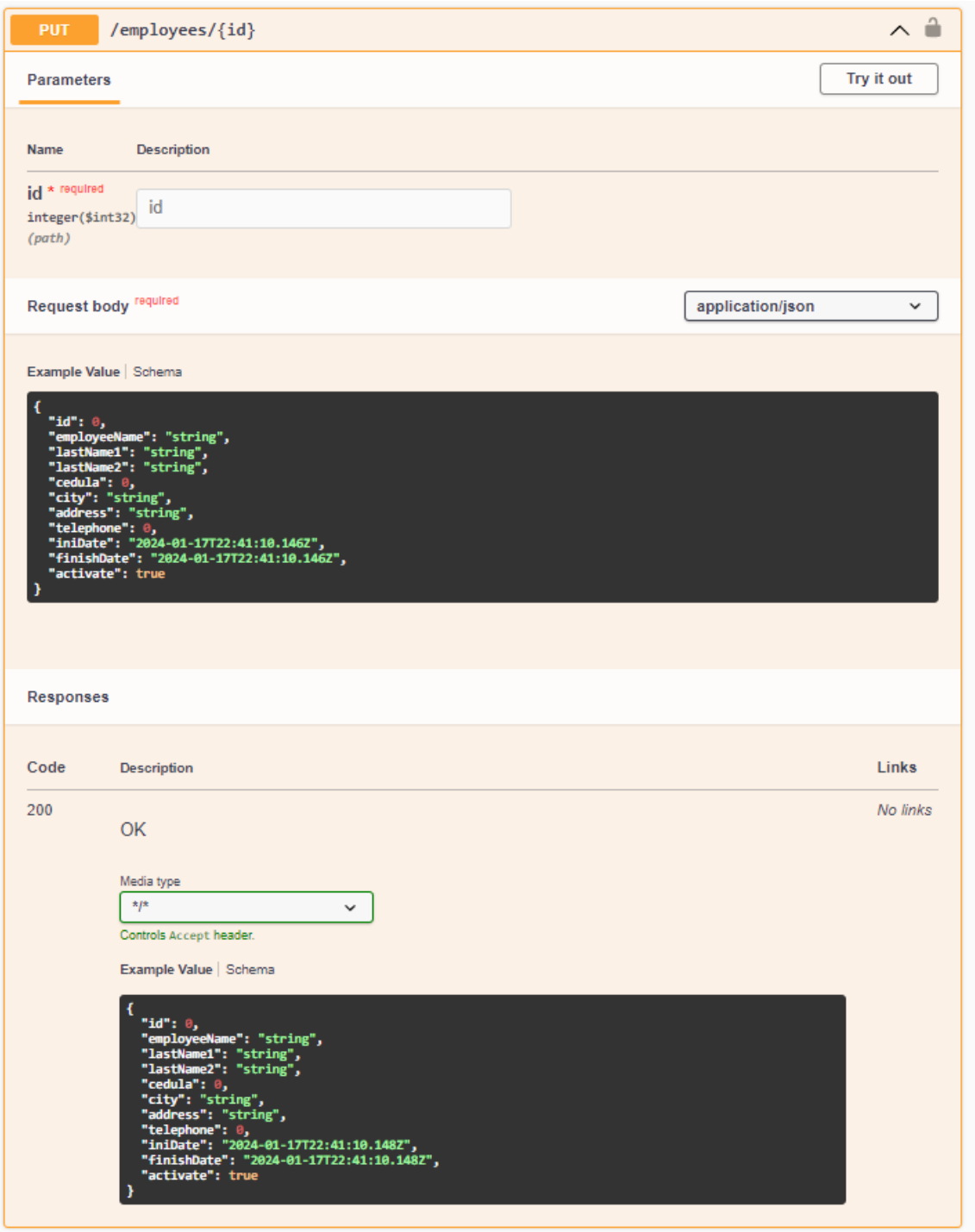
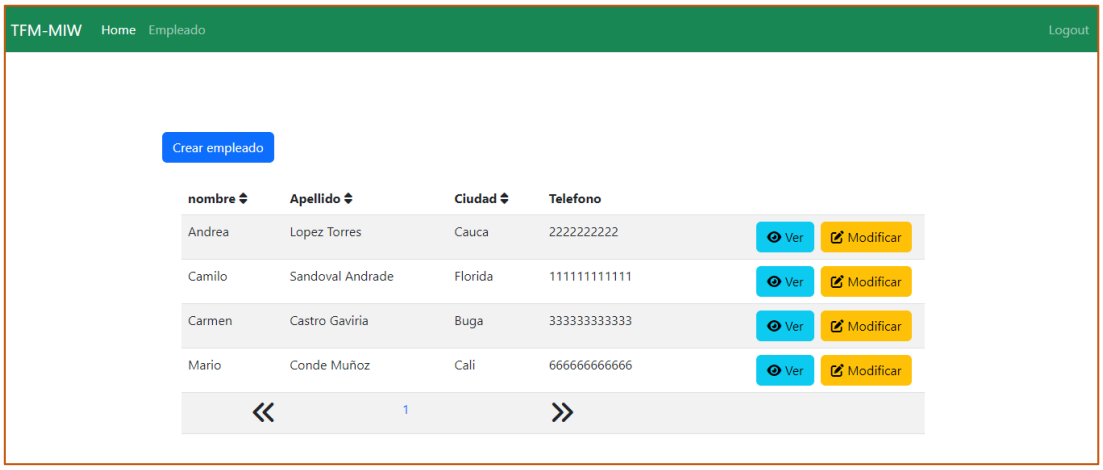


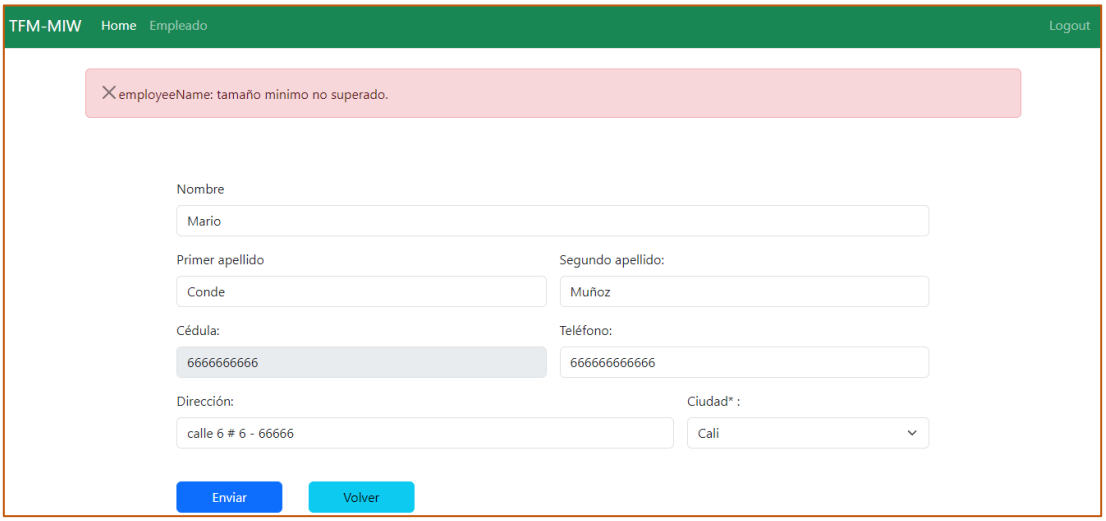
Ilustración 42. PUT/employees/{id}

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para modificar los datos de un empleado.

Al pulsar en la opción Empleado del navbar, una vez se ha autenticado el usuario en el sistema, el usuario tiene la opción de modificar a un empleado pulsando en el botón "Modificar" que se encuentra a la derecha de cada empleado de la lista.

En la siguiente imagen vemos el formulario que se le proporciona al usuario al pulsar en el botón "Modificar" para introducir los datos del empleado que quiere modificar. La vista del formulario con la información del empleado tiene deshabilitada la información que no se puede modificar y contiene un botón de volver que lleva al usuario al listado de empleados; así como también indica al usuario los campos que no cumplen los requisitos mínimos en el formulario y finalmente al pulsar en "Enviar" se muestra una ventana de confirmación al usuario para realizar la modificación del empleado.





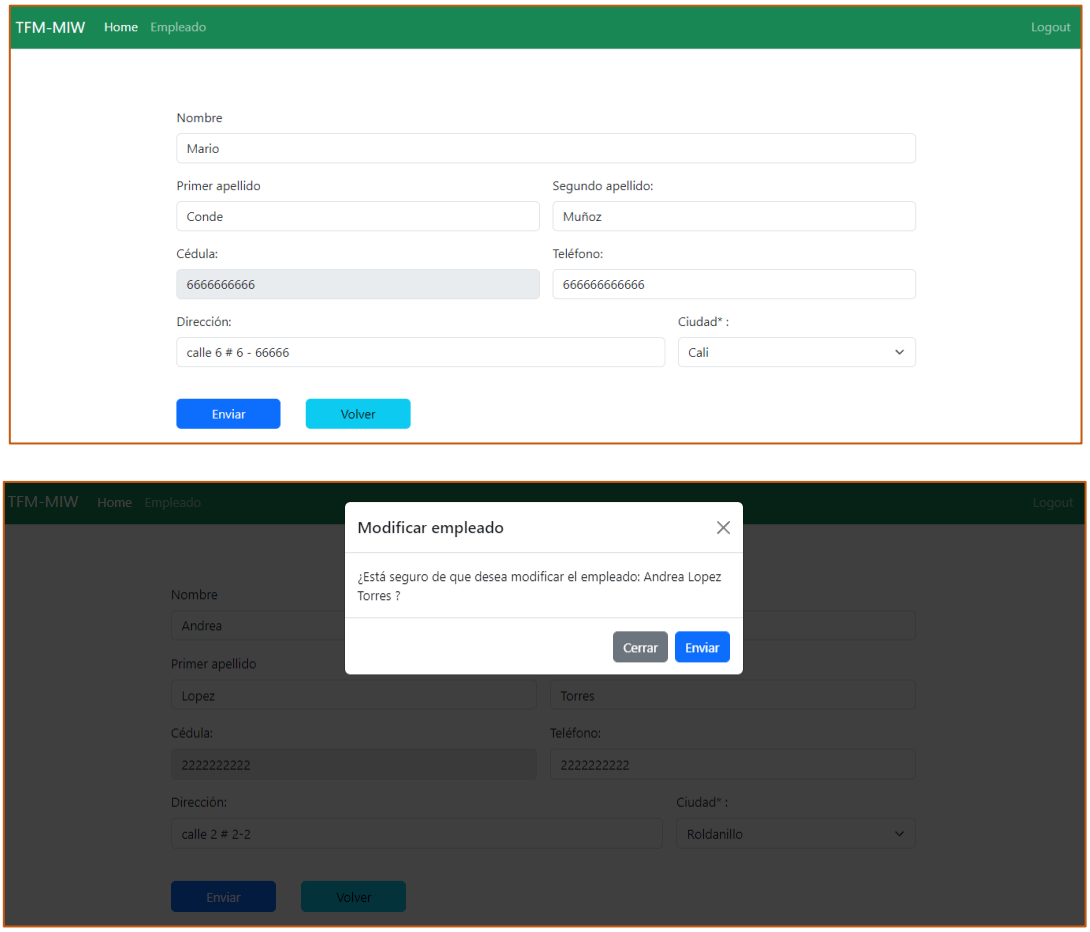


Ilustración 43. Pantallas modificar empleado

8.2.1.2.4 Historia de Usuario Eliminar empleado

PUT/employees/disable/{id}:

Permite realizar la baja lógica de un empleado en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador del empleado y se debe enviar de forma obligatoria para verificar que dicho empleado existe en el sistema antes realizar el proceso de baja lógica.

La información correspondiente al atributo actívate se le asigna el valor false y al atributo finishDate se le asigna la fecha del sistema en el momento que se accede al endpoint para posteriormente realizar el proceso de modificación del empleado.

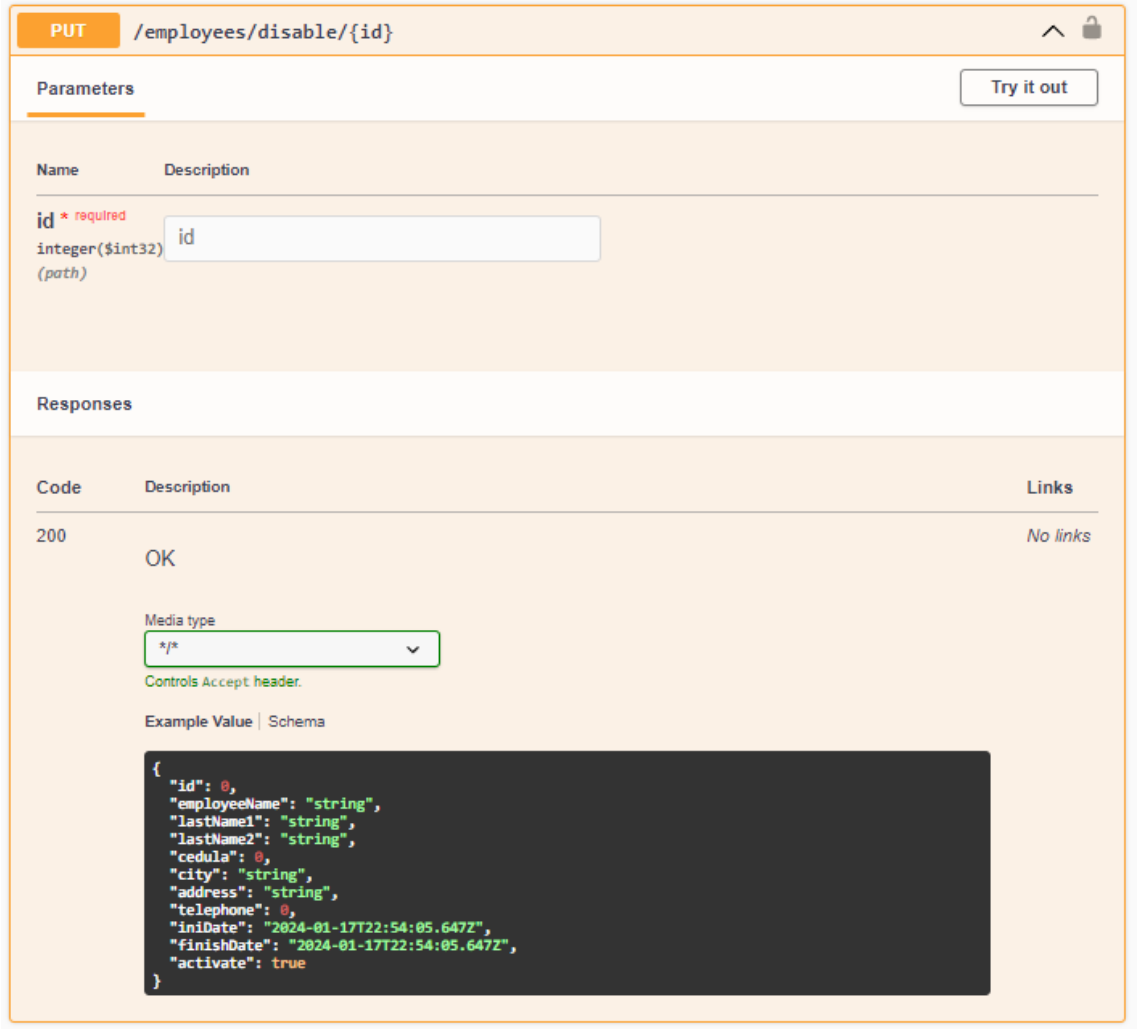
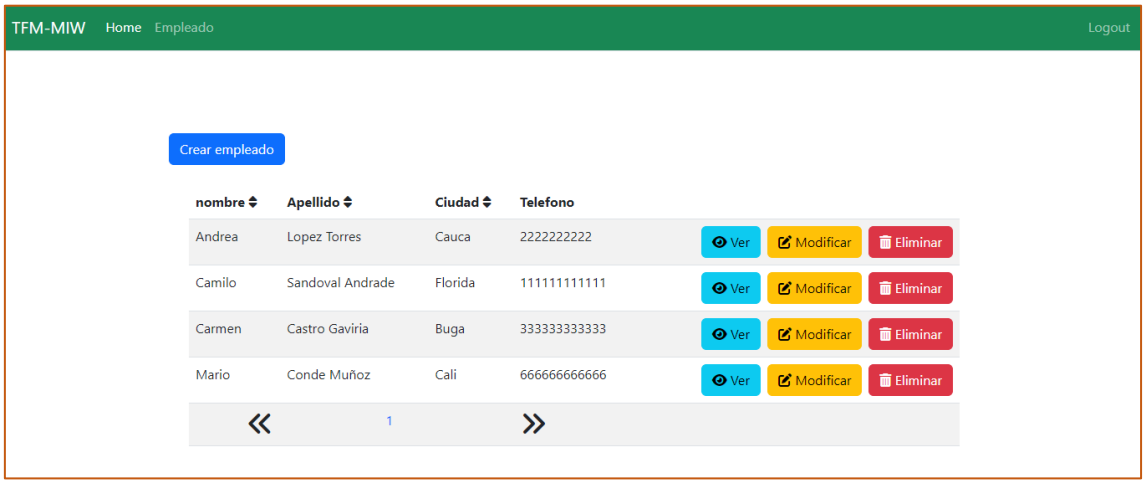


Ilustración 44. PUT/employees/disable/{id}

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para realizar la baja lógica de un empleado.

Al pulsar en la opción Empleado del navbar, una vez se ha autenticado el usuario en el sistema, éste tiene la opción de dar de baja a un empleado pulsando el botón “Eliminar” que se encuentra a la derecha de cada empleado de la lista.

Cuando se hace click en el botón “Eliminar” se solicita la confirmación al usuario para dar de baja al empleado seleccionado.



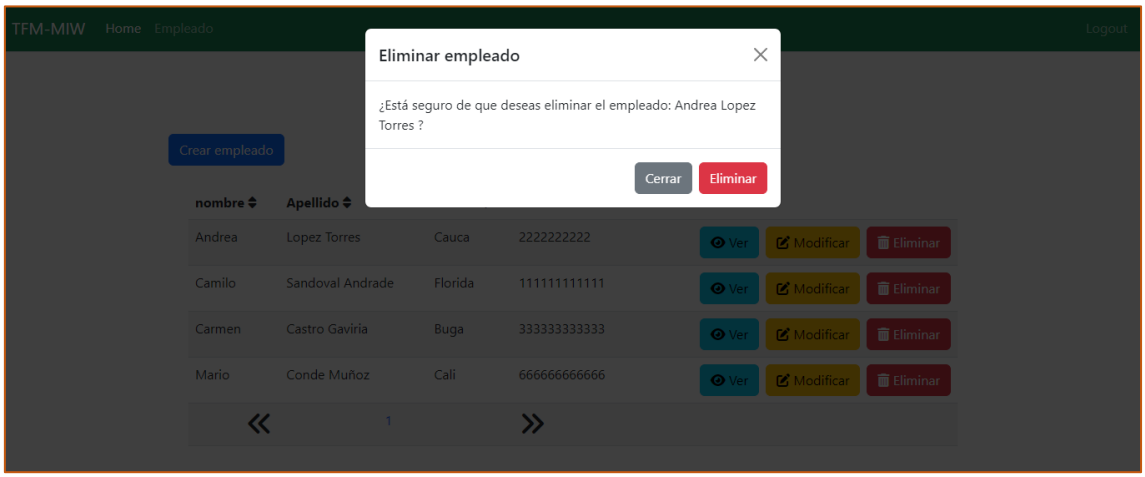


Ilustración 45. Pantallas eliminar empleado

8.2.1.2.4.1 Sprint Retrospective

Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no.

En este proyecto al ser individual el alumno tomará todos los roles dentro del equipo.

Sprint Burndown

A continuación se presenta el gráfico de burdown del sprint. Como se puede apreciar, el desarrollo de este sprint se ha realizado en el tiempo estimado, en cada sprint que se termina se intenta aprender de sprints anteriores para estimar con más realidad las

tareas planificadas en el sprint aunque se ha mejorado bastante la estimación hay que seguir trabajando en ello e intentar mejorarla.

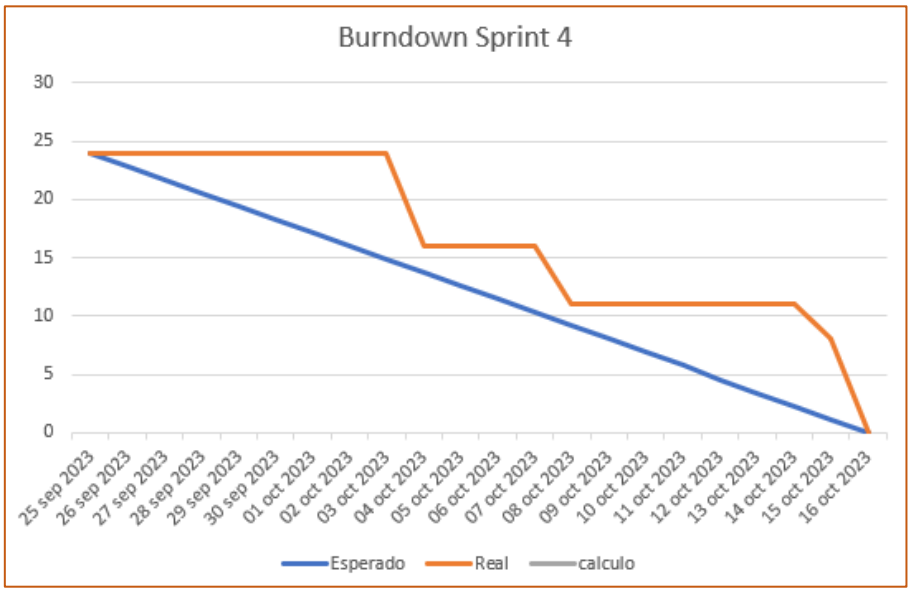


Ilustración 46. Burndown Sprint 2 R2

Para desarrollar el análisis del sprint retrospective desarrollaremos los siguientes apartados en base al desarrollo del Sprint:

- ¿Qué cosas han funcionado bien?
  Se ha conseguido terminar el Sprint con todas las tareas asignadas a éste.
- ¿Cuáles hay que mejorar?
  El tiempo estimado para algunas tareas del Sprint no ha sido correcto ya que han necesitado más esfuerzo del estimado mientras que otras fueron estimadas pensando que necesitaban más esfuerzo y finalmente no lo han necesitado
  No se han terminado los test unitarios del código desarrollado hasta este punto del desarrollo de la aplicación; por lo tanto no se ha ampliado la cobertura para llegar al 80% que es lo aconsejable y hay algunos cod smells que hay que corregir según el informe de sonar.

- ¿Qué cosas quiere probar hacer en la siguiente iteración?
  Estimar mejor las Historias de usuario y sus tareas. Ampliar un poco la cobertura del código.
- ¿Qué ha aprendido?
  Se van reforzando los conocimientos en desarrollo frontend necesitando menos tiempo para el desarrollo del proyecto.
- Cuáles son los problemas que podrían impedirle progresar adecuadamente.
  La incorrecta estimación de las tareas a realizar puesto que el tiempo es limitado.

8.3 Release 3

Features que se desarrollarán en esta release:

Ilustración 47. Features de la release 3

Historias de usuario que se desarrollaran en esta release:

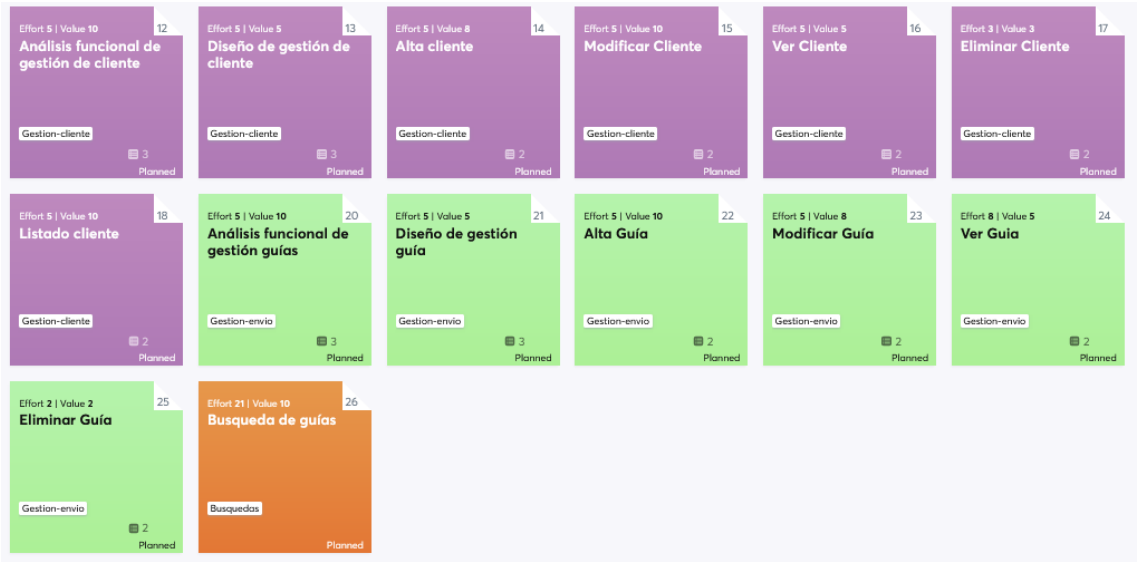


Ilustración 48. Historias de usuario de la release 3

8.3.1 Desarrollo de Sprints Release 3

8.3.1.1 Sprint 1 R3

Sprint planing: 16-10-2023 al 05-11-2023

En este sprint se van a desarrollar las historias de usuario para el análisis y diseño de la gestión de clientes, así como las operaciones relacionadas con la gestión de clientes.

A continuación se detallarán las tareas correspondientes a las historias de usuario del backlog implicadas en este sprint.

Identificador	S5T26	**Historia de usuario**	Análisis funcional de gestión de clientes
Nombre	Diagrama de casos de uso de gestión cliente		
Descripción	Generar el diagrama de casos de uso necesario para la gestión de clientes		
Puntos de historia	1	**Prioridad**	Alta

Identificador	S5T27	**Historia de usuario**	Análisis funcional de gestión de clientes
Nombre	Diagrama entidad relación de gestión cliente		
Descripción	Generar el diagrama de entidad relación necesario para la gestión de clientes		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T28	**Historia de usuario**	Análisis funcional de gestión de clientes

Nombre	Diagrama de clases de gestión cliente		
Descripción	Generar el diagrama de clases necesario para la gestión de clientes		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T29	**Historia de usuario**	Diseño de gestión de clientes
Nombre	Paso a tablas gestión clientes		
Descripción	Realizar el diagrama de tablas necesario para la gestión de clientes		
Puntos de historia	1	**Prioridad**	Alta

Identificador	S5T30	**Historia de usuario**	Diseño de gestión de cliente
Nombre	Paso a tablas gestión clientes		
Descripción	Realizar el diagrama de tablas necesario para la gestión de clientes		
Puntos de historia	1	**Prioridad**	Alta

Identificador	S5T31	**Historia de usuario**	Diseño de gestión de cliente
Nombre	Diagrama de secuencia de gestión cliente		
Descripción	Realizar el diagrama de secuencia necesario para la gestión de clientes		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S5T32	**Historia de usuario**	Listado clientes
Nombre	Crear la lógica necesaria en la parte backend de la aplicación para obtener el listado de los clientes activos de la empresa y exponer un servicio con paginación que devuelva la lista obtenida.		
Descripción			
Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T33	**Historia de usuario**	Listado clientes
Nombre	Crear listado de clientes en la parte cliente		
Descripción	En Angular, crear el componente cliente y agregar la opción clientes en el navbar, el cual mostrará una tabla con paginación que contiene la lista de clientes activos de la empresa. Estos clientes activos se obtendrán consumiendo un servicio expuesto por la parte backend de la aplicación.		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S5T34	**Historia de usuario**	Alta cliente
Nombre	Crear cliente en servidor		
Descripción	Agregar la lógica necesaria para crear clientes en el servidor y exponer el servicio		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T35	**Historia de usuario**	Alta cliente
Nombre	Agregar la opción crear cliente en la parte cliente		
Descripción	Agregar a la vista de listado de clientes la opción de crear empleado, siendo esta un botón en la parte superior izquierda. Agregar la lógica necesaria para la validación del formulario y el		

	envío a través de la api al servicio correspondiente expuesto en el servidor		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T36	**Historia de usuario**	Alta cliente
Nombre	Crear listado de clientes en la parte cliente		
Descripción	En Angular, crear el componente cliente y agregar la opción clientes en el navbar, el cual mostrará una tabla con paginación que contiene la lista de clientes activos de la empresa. Estos clientes activos se obtendrán consumiendo un servicio expuesto por la api del servidor de la aplicación.		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S5T37	**Historia de usuario**	Ver cliente
Nombre	Crear obtener cliente por id en la parte servidor		
Descripción	Agregar la lógica necesaria para obtener la información de un cliente por el id en Spring y exponer el servicio.		
Puntos de historia	2	**Prioridad**	Media

Identificador	S5T38	**Historia de usuario**	Ver cliente
Nombre	Agregar la opción ver cliente en la parte cliente		
Descripción	En Angular, agregar en la opción de listado de clientes, en la fila de cada cliente y a la derecha un botón para ver la información completa del cliente; así como también agregar la lógica necesaria para la validación del formulario y el envío a través de la api al servicio correspondiente expuesto en el servidor		

Puntos de historia	3	**Prioridad**	Media

Identificador	S5T39	**Historia de usuario**	Modificar cliente
Nombre	Crear modificar cliente en servidor		
Descripción	Agregar la lógica necesaria para modificar cliente en Spring y exponer el servicio que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	2	**Prioridad**	Media

Identificador	S5T40	**Historia de usuario**	Modificar cliente
Nombre	Agregar la opción modificar cliente en la parte cliente		
Descripción	En Angular, agregar en la opción de listado de clientes, en la fila de cada cliente y a la derecha un botón de modificar; el cual permitirá modificar un cliente. Agregar la lógica necesaria para la validación del formulario y el envío a través de la api al servicio correspondiente expuesto en el servidor		
Puntos de historia	3	**Prioridad**	Media

Identificador	S5T41	**Historia de usuario**	Eliminar cliente
Nombre	Crear deshabilitar cliente en servidor		
Descripción	Agregar la lógica necesaria para realizar el borrado lógico de un cliente en el servidor y exponer el servicio		
Puntos de historia	1	**Prioridad**	Baja

Identificador	S5T42	**Historia de usuario**	Eliminar cliente
Nombre	Agregar opción eliminar cliente en la parte cliente		

Descripción	En Angular, agregar en la opción de listado de clientes, en la fila de cada cliente y a la derecha un botón de eliminar; el cual permitirá deshabilitar un cliente. Agregar la lógica necesaria para la validación del formulario y el envío a través de la api al servicio correspondiente expuesto en el servidor		
Puntos de historia	2	Prioridad	Baja

8.3.1.1.1 Historia de Usuario Análisis funcional de gestión de cliente

8.3.1.1.1.1 Tarea 1. Diagrama de casos de uso de gestión cliente

El diagrama de casos de uso representa la forma en como un usuario (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan.

En el apartado Diagrama de casos de uso de gestión empleado se puede ver la explicación de la generación de los casos de uso.

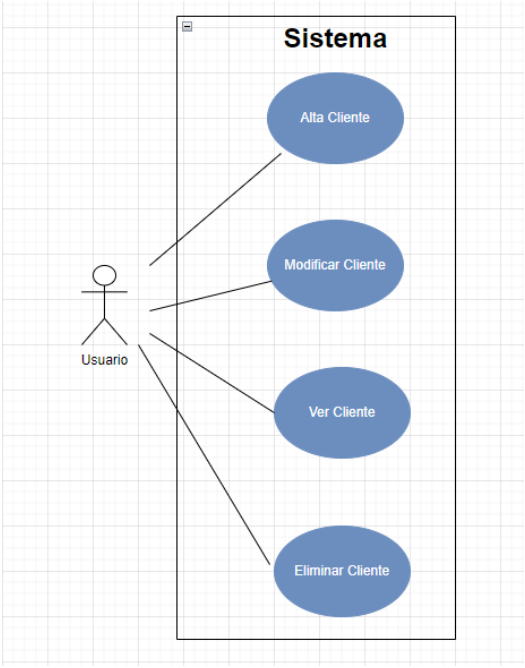


Ilustración 49. Diagrama casos de uso gestión cliente

La descripción de los casos de uso se hará a través de la descripción de las tareas en las que se ve involucrada la gestión de clientes.

8.3.1.1.1.2 Tarea. Diagrama entidad relación de gestión de clientes

Un diagrama o modelo entidad-relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades. Las características del modelo entidad-relación están explicadas en el apartado Diagrama entidad relación de gestión de empleados Diagrama entidad relación de gestión de empleados

A continuación se agregará al modelo entidad-relación existente la parte correspondiente a la gestión de cliente. Se agrega al modelo entidad relación ya existente la entidad cliente quedando con dos entidades sin relación entre ellas.

Ilustración 50. Diagrama entidad relación gestión cliente

8.3.1.1.1.3 Tarea. Diagrama de clases de gestión de clientes

La explicación de qué es y las características de un diagrama de clases se puede ver en el apartado Diagrama de clases de gestión de empleados

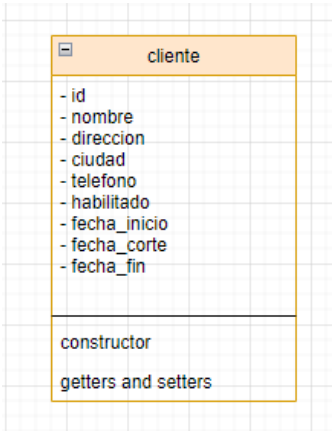


Ilustración 51. Diagrama de clases gestión empleados

8.3.1.1.2 Historia de Usuario Diseño de gestión de cliente

En este apartado se va a desarrollar el diseño de gestión de clientes, el cual ayuda a entender cómo desarrollaremos nuestra aplicación.

8.3.1.1.2.1 Tarea 1. Paso a tablas gestión clientes

La definición y características del paso a tablas está explicado en el apartado Paso a tablas gestión empleados Paso a tablas gestión empleados

Se agregará al modelo de tablas existentes la estructura correspondiente a la gestión de clientes. Las tablas existentes hasta ahora no tienen ninguna relación entre ellas.

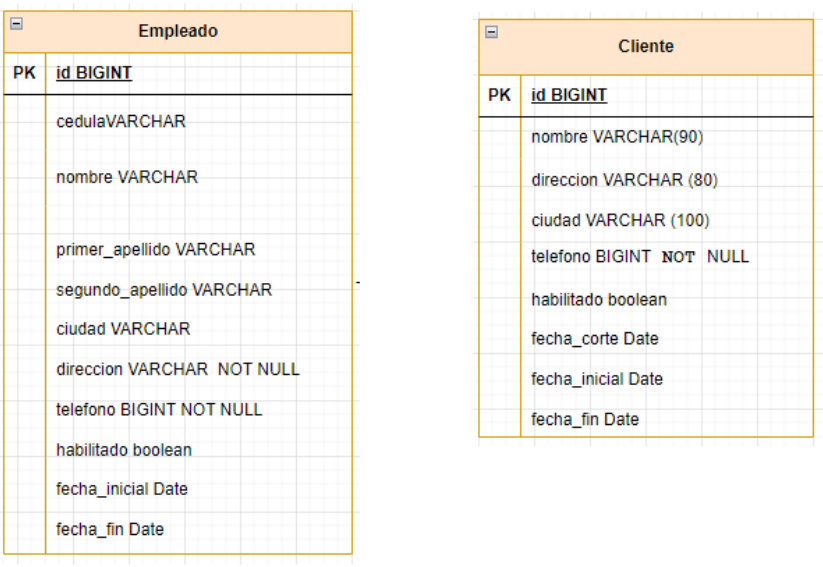


Ilustración 52. Paso a tablas gestión cliente

8.3.1.1.2.2 Tarea 2. BBDD gestión cliente

En cuanto al modelo de datos en este sprint corresponde en agregar las tabla cliente al modelo de datos.

```
CREATE TABLE `empleado` (
 `id` BIGINT NOT NULL AUTO_INCREMENT,
 `cedula` BIGINT NOT NULL,
 `nombre` varchar(255) NOT NULL,
 `primer_apellido` varchar(255) NOT NULL,
```

```
  `segundo_apellido` varchar(255) DEFAULT NULL,
  `direccion` varchar(255) NOT NULL,
  `ciudad` varchar(100) DEFAULT NULL,
  `telefono` BIGINT NOT NULL,
  `habilitado` bit(1) NOT NULL DEFAULT b'1',
  `fecha_inicio` date DEFAULT NULL,
  `fecha_fin` date DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;


CREATE TABLE `cliente` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `nombre` varchar(255) NOT NULL,
  `direccion` varchar(255) NOT NULL,
  `ciudad` varchar(100) DEFAULT NULL,
  `telefono` BIGINT NOT NULL,
  `habilitado` bit(1) NOT NULL default b'1',
  `fecha_corte` INT DEFAULT NULL,
  `fecha_inicio` date DEFAULT NULL,
  `fecha_fin` date DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

8.3.1.1.2.3 Tarea 3. Diagrama de secuencia de gestión cliente

La explicación de qué es y las características de un diagrama de secuencia se puede ver en el apartado Diagrama de secuencia de gestión empleado

Ilustración 53. Diagrama de secuencia de gestión cliente

8.3.1.1.3 Historia de Usuario Listado cliente

8.3.1.1.3.1 Tarea 1. Obtener listado clientes activos paginado en la parte servidor

Endpoint creado en el backend para la obtención de clientes activos con paginación.

GET/customers/activate:

Permite obtener la lista de empleados activos en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición. Los parámetros a enviar corresponden con:

- Page: la página de inicio de carga
- El tamaño de registros por pagina
- El atributo por el que se quiere realizar la ordenación de los registros
- El orden de los registros que se van a visualizar

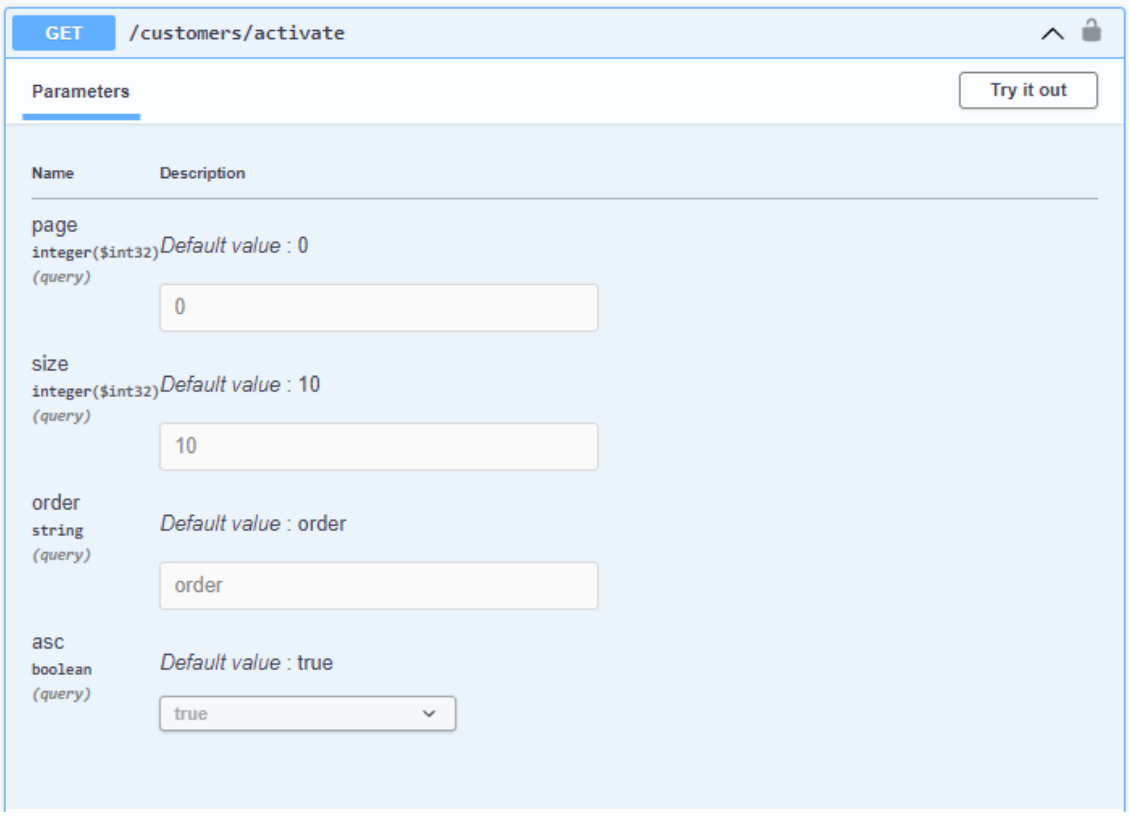


Ilustración 54. GET/customers/activate

8.3.1.1.3.2 Tarea 2. Crear listado de clientes en la parte cliente

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para obtener el listado de los cliente activos con paginación.

Al pulsar en la opción “Cliente” del navbar, una vez se ha autenticado el usuario en el sistema, se puede ver el listado en una tabla paginada con 10 elementos.

En la siguiente imagen se puede ver el resultado del listado.

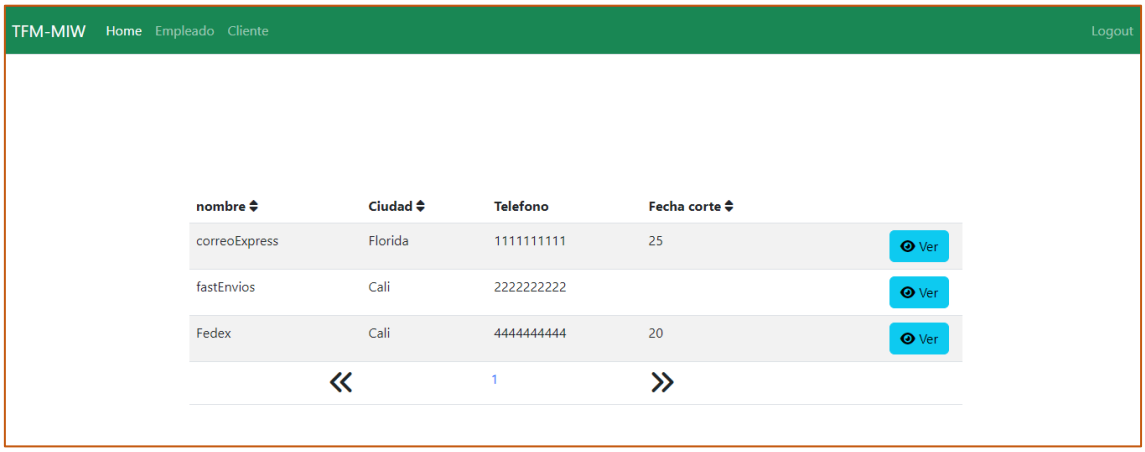


Ilustración 55. Pantalla listado clientes

8.3.1.1.4 Historia de Usuario Alta cliente

8.3.1.1.4.1 Tarea 1. Crear cliente en servidor

Endpoint creado en el backend para la creación de clientes.

POST/customers:

Permite la creación de un cliente en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

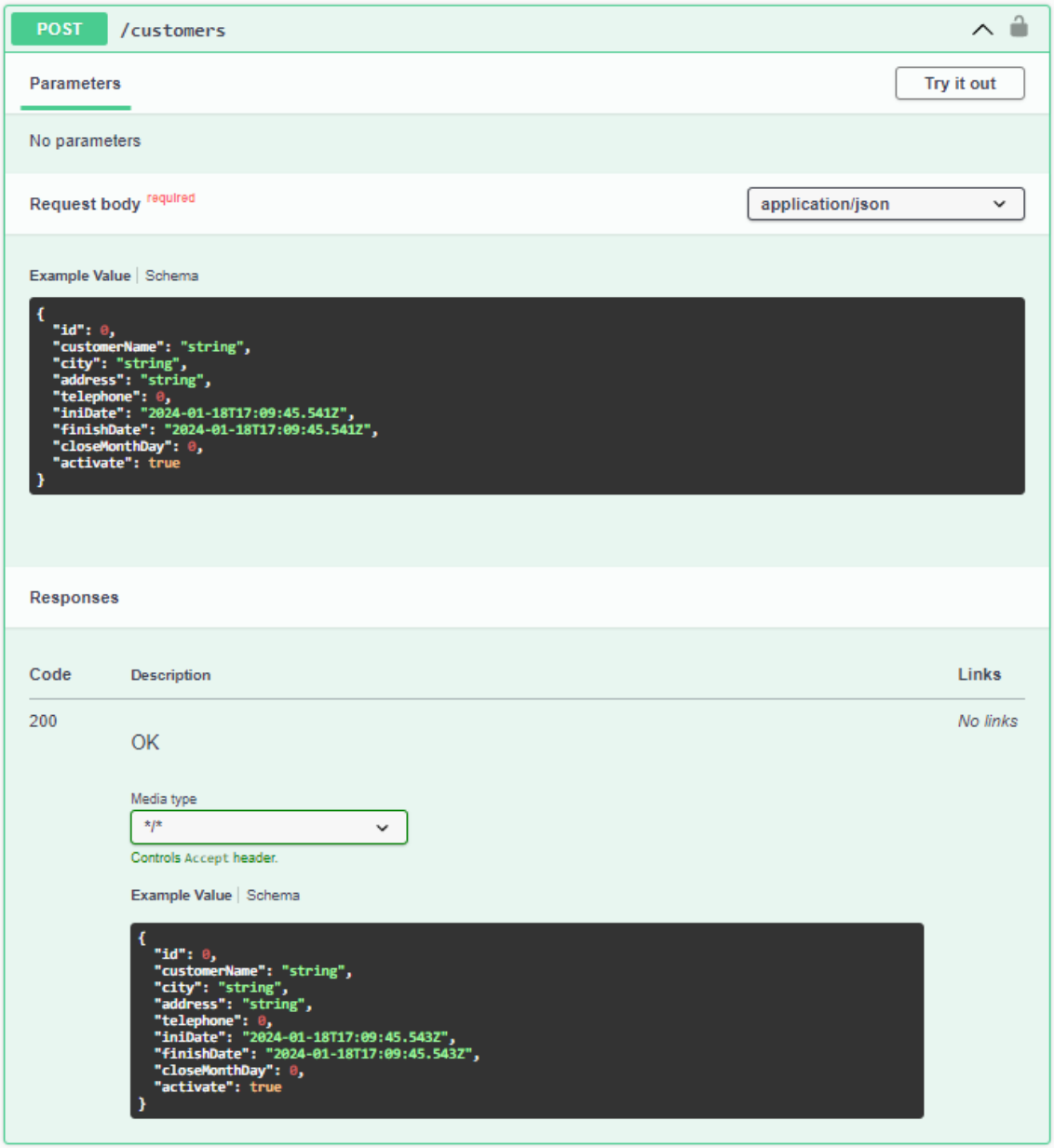


Ilustración 56. POST/customers

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para dar de alta a un cliente.

Al pulsar en la opción Cliente del navbar, una vez se ha autenticado el usuario, se muestra la vista de listado de clientes y en la parte superior izquierda se tiene la opción de crear un nuevo cliente.

En la siguiente imagen vemos el formulario que se le proporciona al usuario al pulsar en el botón “Crear Cliente” para introducir los datos de éste. Enel formulario existe un campo llamado “Dia cierre” el cual indica el día de corte de facturación que tiene la empresa para facturar hasta ese día del mes al cliente. El botón “Enviar” no estará habilitado para ser pulsado hasta que no se hayan introducido todos los datos que son obligatorios en el formulario y tampoco se habilitará si no se cumplen los requisitos de cada campo, en el caso de que no se cumplan los requisitos de cada campo se mostrará

un mensaje al usuario indicándolo. También se ha agregado un botón para poner todos los campos del formulario en blanco si el usuario lo desea.

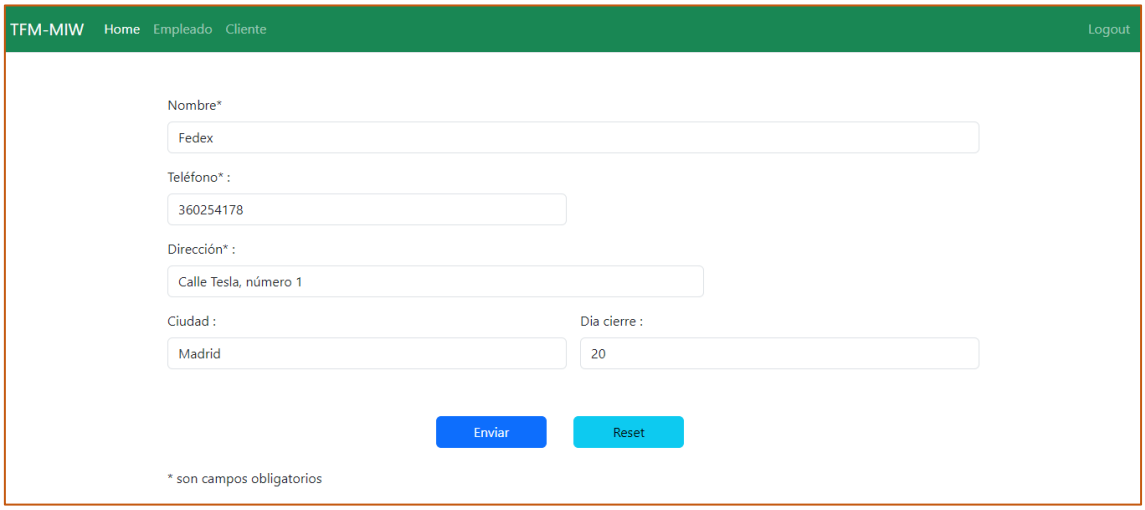


Ilustración 57. Pantalla formulario crear cliente

8.3.1.1.5 Historia de Usuario Ver Cliente

8.3.1.1.5.1 Tarea 1. Crear obtener cliente en la parte servidor

Endpoint creado en el backend para la realizar la búsqueda de un cliente por su id.

GET/customers/{id}:

Permite obtener la información de un cliente realizando la búsqueda por su id en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador del cliente y se debe enviar de forma obligatoria.

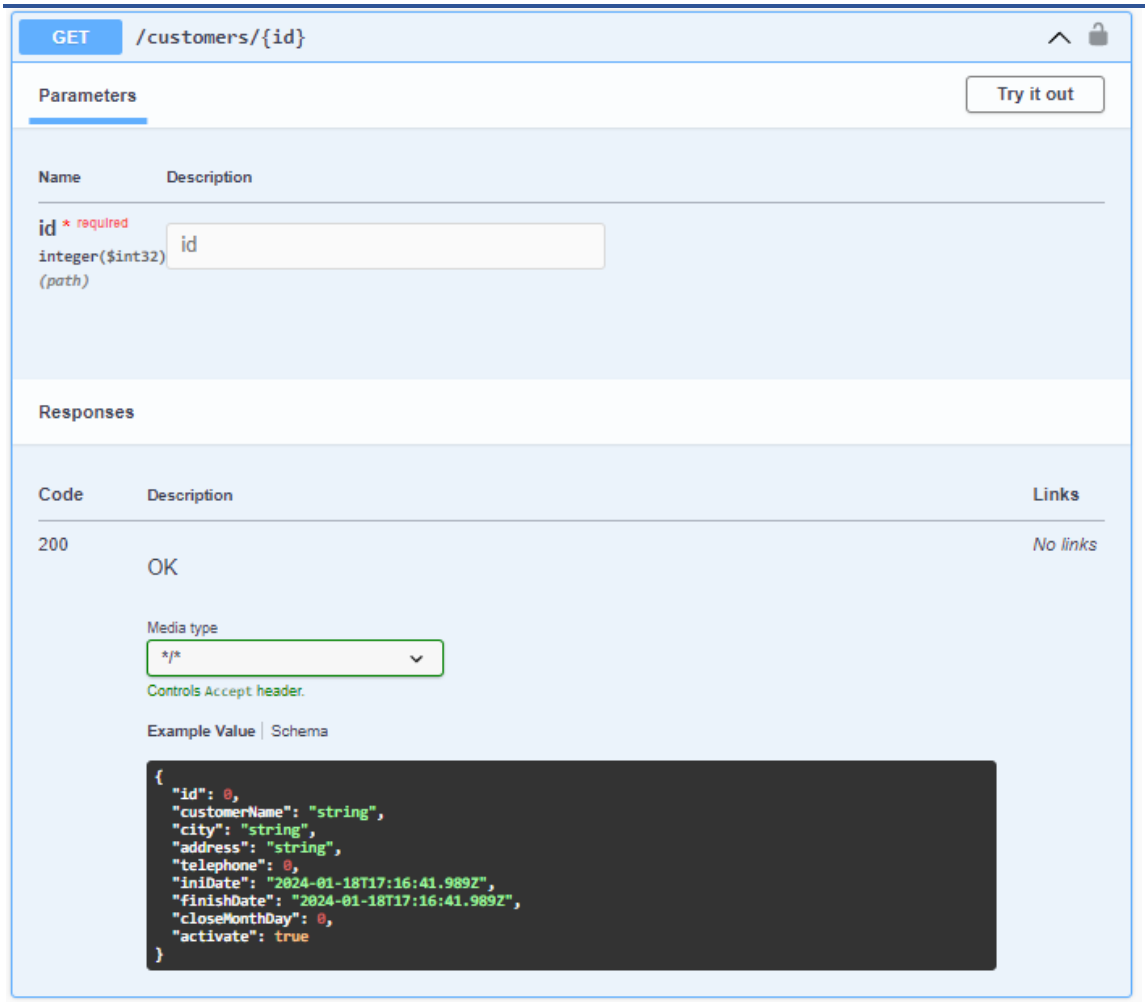


Ilustración 58. GET/customers/{id}

Este es el resultado de la generación del código necesario tanto en FrontEnd y BackEnd para ver los datos detallados de un cliente.

Al pulsar en la opción Cliente del navbar, una vez se ha autenticado el usuario, se carga la lista de clientes en la cual tiene la opción de ver los datos detallados de cada cliente pulsando el botón “Ver” que se encuentra a la derecha de cada cliente de la lista, la vista detallada de la información de cada cliente muestra la información deshabilitada en modo solo lectura y contiene un botón de volver que lleva al usuario al listado de clientes.

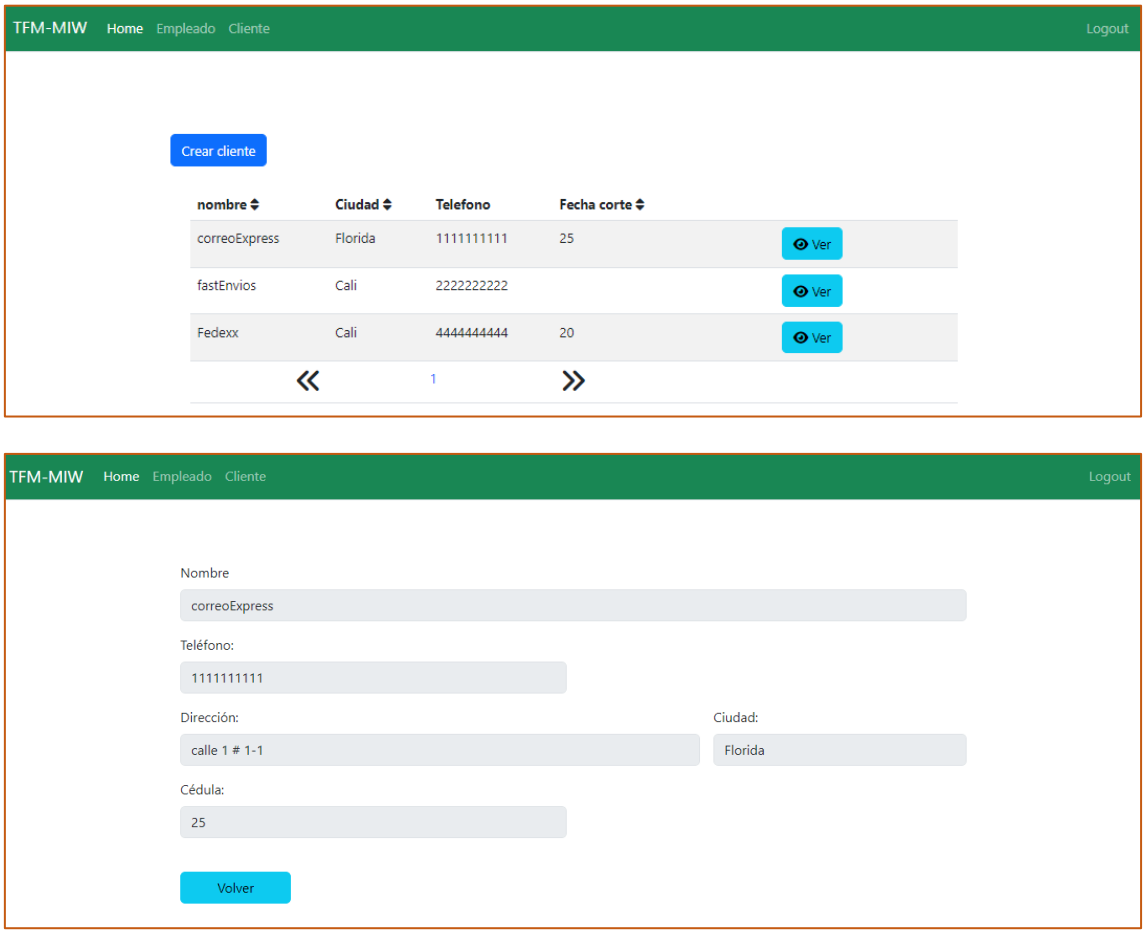


Ilustración 59. Pantallas ver cliente

8.3.1.1.6 Historia de Usuario Modificar Cliente

8.3.1.1.6.1 Tarea 1. Crear modificar cliente en servidor

Endpoint creado en el backend para la modificación de un cliente.

PUT/customers/{id}:

Permite la modificación de la información de un cliente en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador del cliente y se debe enviar de forma obligatoria para verificar que dicho cliente existe en el sistema antes realizar el proceso de modificación.

La información del objeto enviado en el body corresponde a la información del cliente con los datos que se quieren modificar siendo obligatorio el envío de éste.

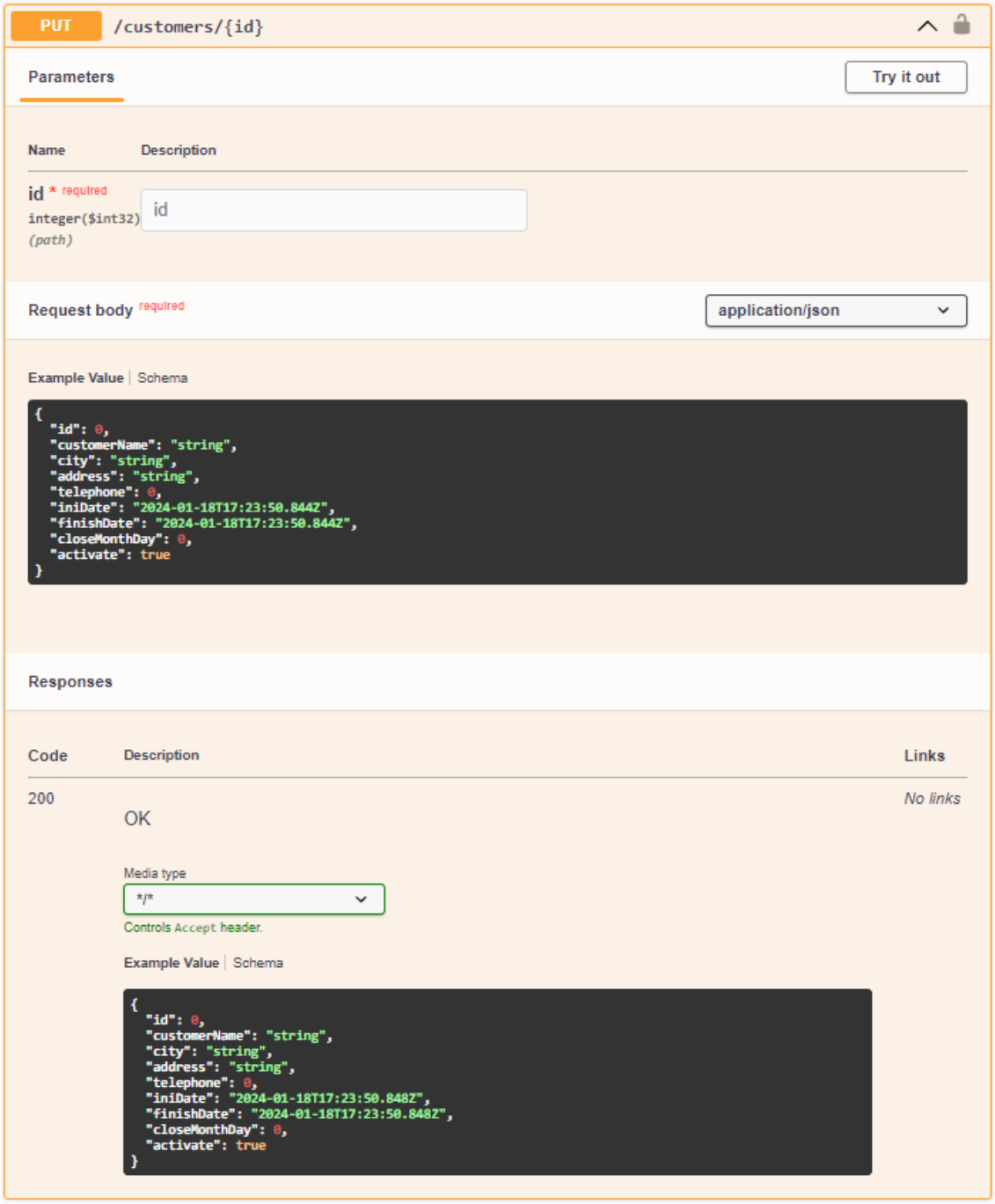
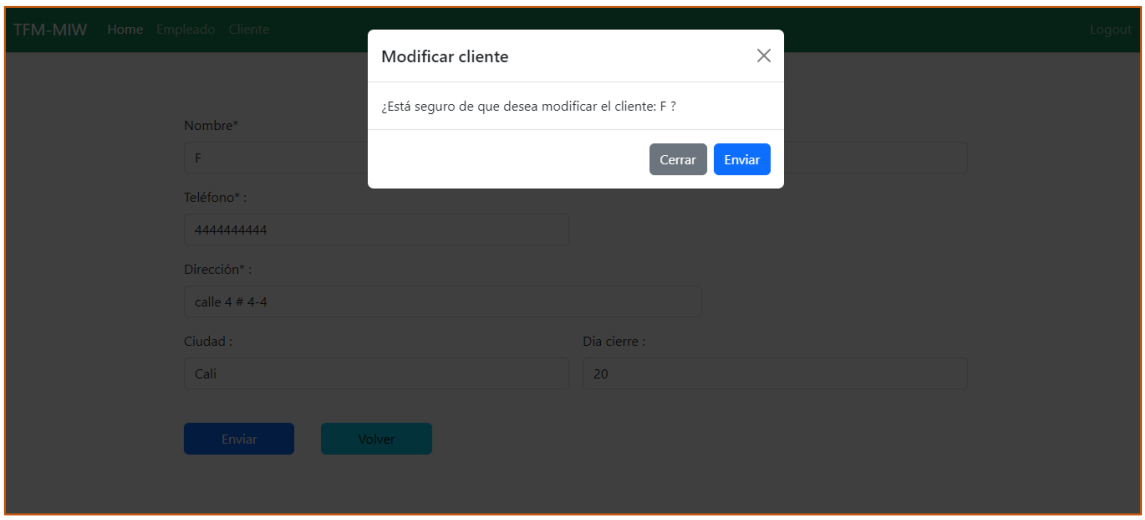


Ilustración 60. PUT/customers/{id}

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para modificar los datos de un cliente.

Al pulsar en la opción Cliente del navbar, una vez se ha autenticado el usuario en el sistema, el usuario tiene la opción de modificar a un cliente pulsando en el botón "Modificar" que se encuentra a la derecha de cada cliente de la lista.

En la siguiente imagen vemos el formulario que se le proporciona al usuario al pulsar en el botón “Modificar” para introducir los datos del cliente que quiere modificar. La vista del formulario con la información del cliente tiene deshabilitada la información que no se puede modificar y contiene un botón de volver que lleva al usuario al listado de clientes; así como también indica al usuario los campos que no cumplen los requisitos mínimos en el formulario.



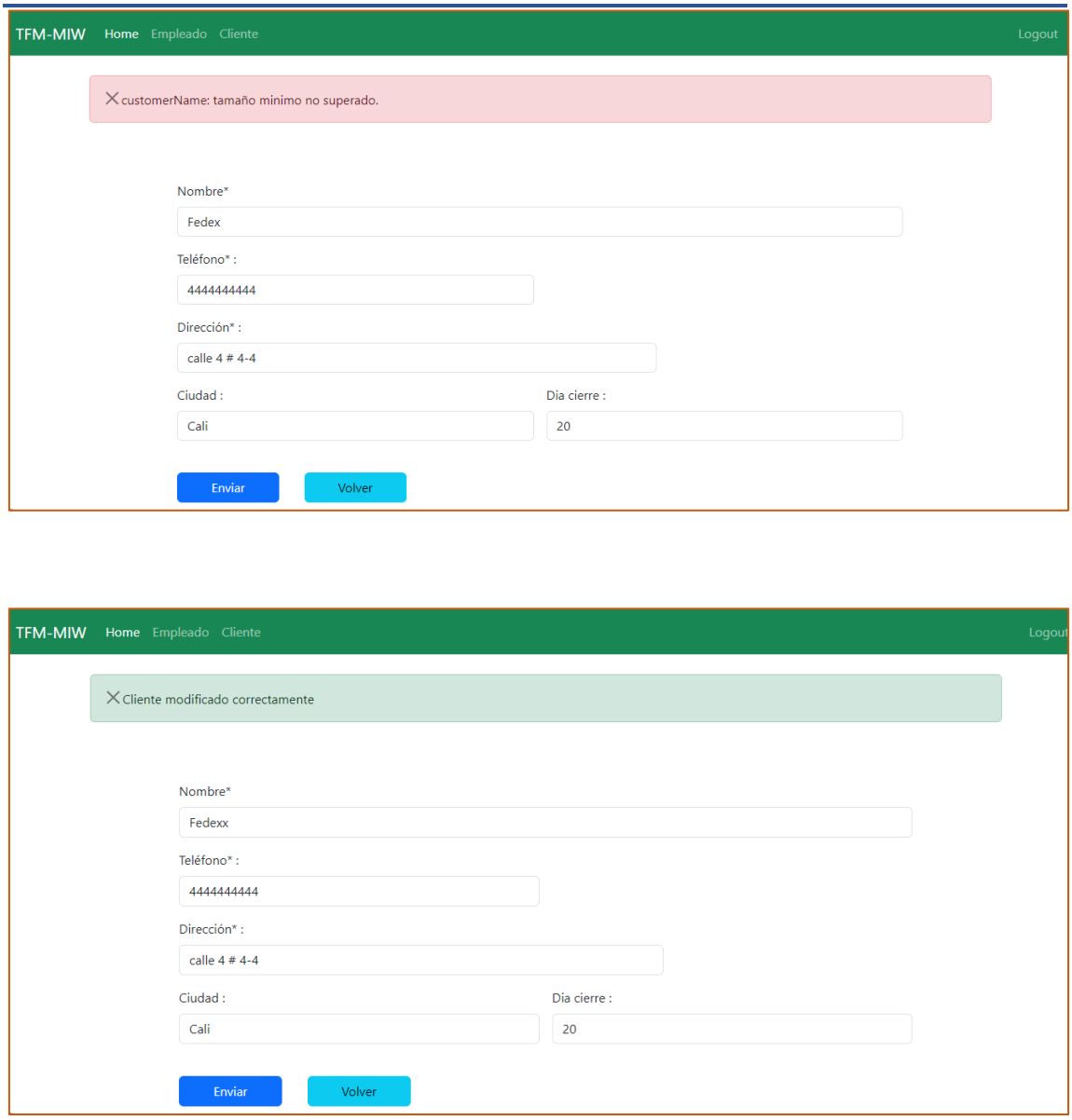


Ilustración 61. Pantallas modificar cliente

8.3.1.1.7 Historia de Usuario Eliminar Cliente

8.3.1.1.7.1 Tarea 1. Crear deshabilitar cliente en servidor

PUT/customers/disable/{id}:

Permite realizar la baja lógica de un cliente en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador del cliente y se debe enviar de forma obligatoria para verificar que dicho cliente existe en el sistema antes realizar el proceso de baja lógica.

La información correspondiente al atributo actívate se le asigna el valor false y al atributo finishDate se le asigna la fecha del sistema en el momento que se accede al endpoint para posteriormente realizar el proceso de modificación del cliente.

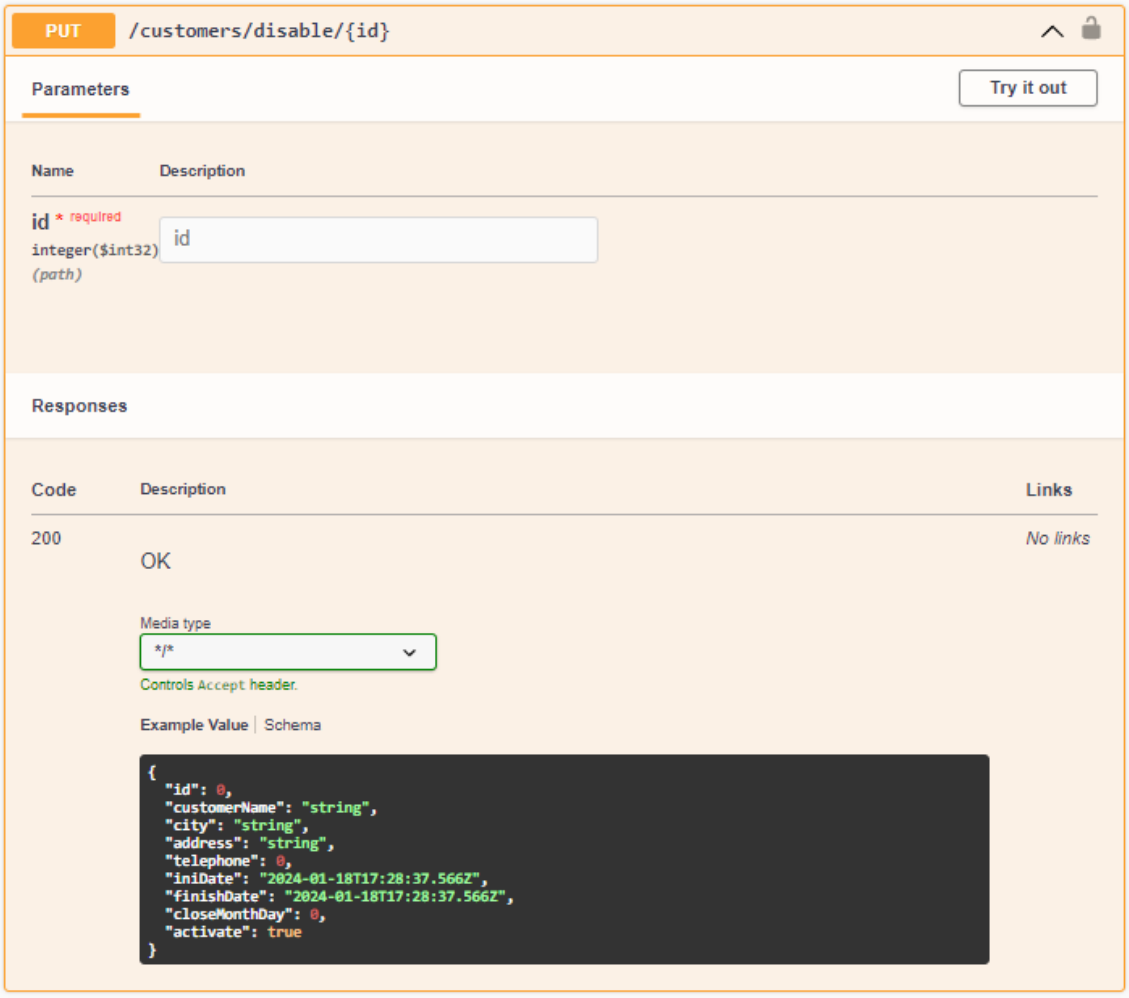


Ilustración 62. PUT/customers/disable/{id}

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para realizar la baja lógica de un cliente.

Al pulsar en la opción Cliente del navbar, una vez se ha autenticado el usuario en el sistema, éste tiene la opción de dar de baja a un cliente pulsando el botón “Eliminar” que se encuentra a la derecha de cada cliente de la lista.

Cuando se hace click en el botón “Eliminar” se solicita la confirmación al usuario para eliminar al cliente seleccionado.

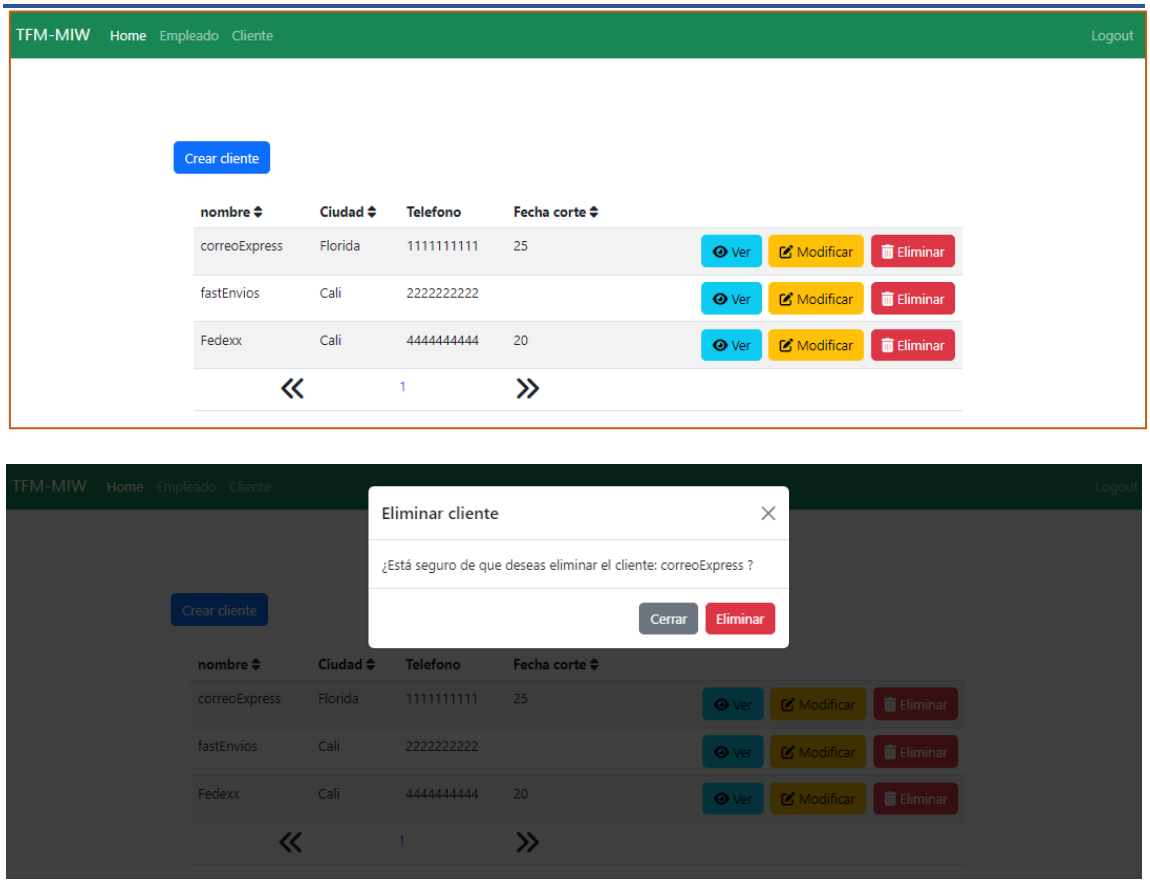


Ilustración 63. Pantallas eliminar cliente

8.3.1.1.7.2 Sprint Retrospective

Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no.

En este proyecto al ser individual el alumno tomará todos los roles dentro del equipo.

Sprint Burndown

A continuación se presenta el gráfico de burdown del sprint .Como se puede apreciar, el desarrollo de este sprint se ha realizado en el tiempo estimado, en cada sprint que se termina se intenta aprender de sprints anteriores y se consigue estimar con más realidad las tareas planificadas en el sprint.

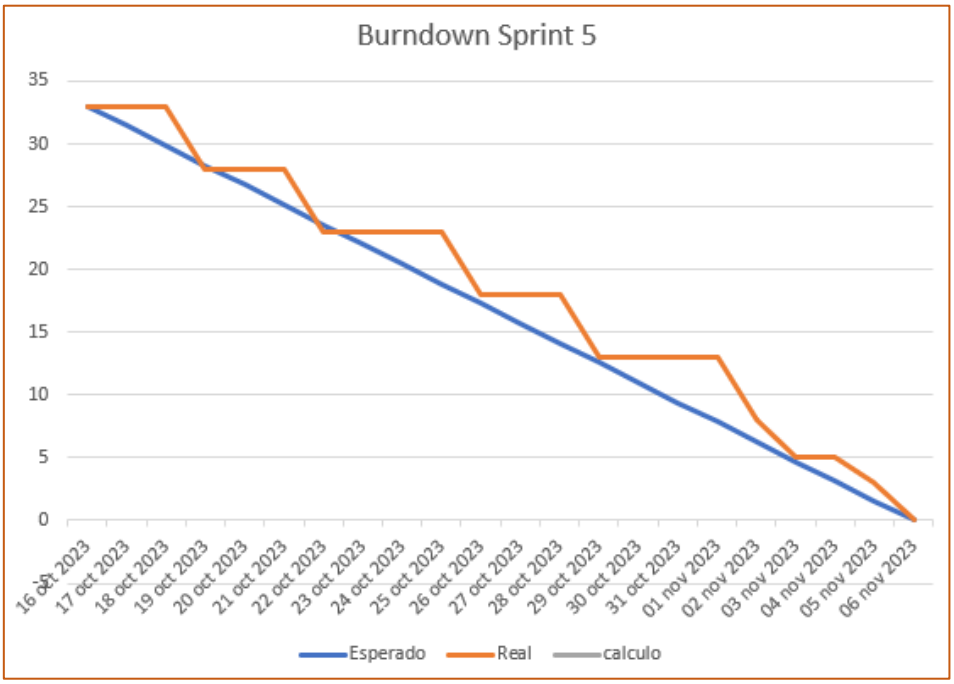


Ilustración 64. Burndown Sprint 1 R3

Para desarrollar el análisis del sprint retrospective desarrollaremos los siguientes apartados en base al desarrollo del Sprint:

- ¿Qué cosas han funcionado bien?
  Se ha conseguido terminar el Sprint con todas las tareas asignadas a éste.
- ¿Cuáles hay que mejorar?
  El tiempo estimado para algunas tareas del Sprint no ha sido correcto ya que han necesitado más esfuerzo y ha habido demasiado estrés para conseguir llegar a la entrega en la fecha estimada. No se ha podido utilizar correctamente el environment para obtener los valores de las variables en ellos en Angular; motivo por el cual se ha tenido que agregar en cada servicio de componente una variable con la url de los endpoints necesarios. Se siguen sin terminar los test unitarios del codigo desarrollado hasta este punto del desarrollo de la aplicación; por lo tanto no se ha ampliado la cobertura para llegar al 80% que es lo aconsejable ni se han arreglado los cod smells que hay que corregir según el informe de sonar.
- ¿Qué cosas quiere probar hacer en la siguiente iteración?
  Estimar mejor las Historias de usuario y sus tareas. Se sigue teniendo deuda técnica con respecto a la cobertura del código en el servidor que hay que solucionar y no se quiere dejar pasar más aunque se esté trabajando en entorno de desarrollo y no se esté desplegando en producción.
- ¿Qué ha aprendido?

Se ha van reforzando los conocimientos en desarrollo front-end, siendo más fácil el desarrollo en la parte cliente del proyecto.

- Cuáles son los problemas que podrían impedirle progresar adecuadamente.
  La incorrecta estimación de las tareas a realizar puesto que el tiempo es limitado.
  La deuda técnica.

8.3.1.2 Sprint 2 R3

Sprint planing: 06-11-2023 al 26-11-2023

En este sprint se van a desarrollar las historias de usuario para el análisis y diseño de la gestión de guías, así como las operaciones relacionadas con la gestión de guías.

A continuación se detallarán las tareas correspondientes a las historias de usuario del backlog implicadas en este sprint.

Identificador	S5T43	**Historia de usuario**	Análisis funcional de gestión guías
Nombre	Diagrama de casos de uso de gestión guía		
Descripción	Generar el diagrama de casos de uso necesario para la gestión de guías		
Puntos de historia	1	**Prioridad**	Alta

Identificador	S5T44	**Historia de usuario**	Análisis funcional de gestión guías
Nombre	Generar el diagrama de entidad relación necesario para la gestión de guía		
Descripción	Generar el diagrama de entidad relación necesario para la gestión de guías		

Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T45	**Historia de usuario**	Análisis funcional de gestión guías
Nombre	Diagrama de clases de gestión guías		
Descripción	Generar el diagrama de clases necesario para la gestión de guías		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T46	**Historia de usuario**	Diseño de gestión guía
Nombre	Paso a tablas gestión guías		
Descripción	Realizar el diagrama de tablas necesario para la gestión de guías		
Puntos de historia	1	**Prioridad**	Alta

Identificador	S5T47	**Historia de usuario**	Diseño de gestión guía
Nombre	BBDD gestión guía		
Descripción	Agregar a la base de datos la parte de estructura de gestión de guías correspondiente		
Puntos de historia	1	**Prioridad**	Alta

Identificador	S5T48	**Historia de usuario**	Diseño de gestión guía
Nombre	Diagrama de secuencia de gestión guía		

Descripción	Realizar el diagrama de secuencia necesario para la gestión de guías		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S5T49	**Historia de usuario**	Alta guía
Nombre	crear guía en servidor		
Descripción	Agregar la lógica necesaria para crear una guía en Spring y exponer el servicio que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	2	**Prioridad**	Alta

Identificador	S5T50	**Historia de usuario**	Alta guía
Nombre	Agregar la opción crear guía en la parte cliente		
Descripción	Agregar la opción "Guía" en el navbar, al pulsar en esta opción se debe cargar una página en blanco en la que añadirá en la parte superior izquierda un botón "Agregar guía" que al pulsarlo permita rellenar un formulario con los datos de una guía. Está vista tendrá dos botones "Enviar" y "Reset" que permitirán enviar los datos para dar de alta y dejar vacío de nuevo el formulario respectivamente. Al pulsar el botón "Enviar" se debe validar los datos de la guía, en caso de que no sean correctos se debe mostrar un mensaje indicándolo. Si los datos son correctos se debe enviar los datos de la nueva guía a través de un servicio expuesto por la parte backend de la aplicación		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S5T51	**Historia de usuario**	Búsqueda de guía parámetros no fecha
Nombre	Agregar búsqueda guía por id y estado en servidor		

Descripción	Agregar la lógica necesaria para buscar una guía en el servidor y exponer los servicios que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	5	**Prioridad**	Alta

Identificador	S5T52	**Historia de usuario**	búsqueda de guía parámetros no fecha
Nombre	Agregar búsqueda guía por id y estado en cliente		
Descripción	Agregar en el cliente la lógica necesaria para buscar una guía por id y estado a través de los servicios expuestos en el servidor. La información de la guía se debe mostrar en una tabla en la página de inicio de guías únicamente si se ha buscado una guía y se ha encontrado, si no es así la tabla de información de guías no se mostrará.		
Puntos de historia	8	**Prioridad**	Alta

8.3.1.2.1 Historia de Usuario Análisis funcional de gestión guía

8.3.1.2.1.1 Tarea 1. Diagrama de casos de uso de gestión guías

El diagrama de casos de uso representa la forma en como un usuario (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan. En el apartado Diagrama de casos de uso de gestión empleado se puede ver la explicación de la generación de los casos de uso.

La descripción de los casos de uso se hará a través de la descripción de las tareas en las que se ve involucrada la gestión de clientes.

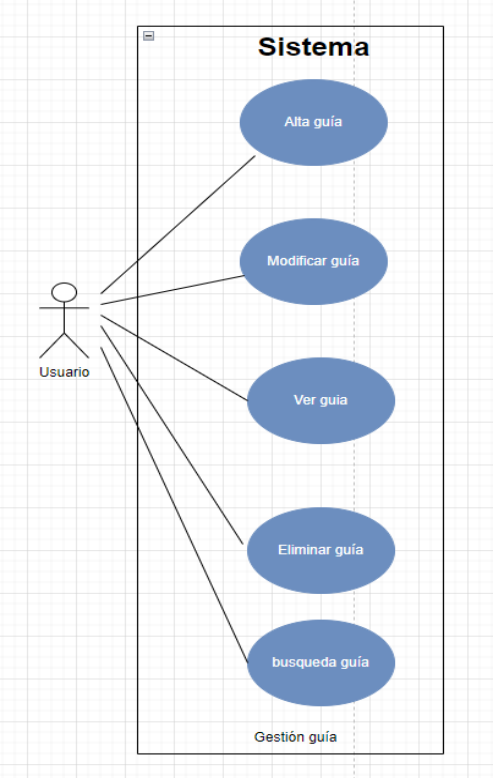


Ilustración 65. Diagrama casos de uso gestión guía

8.3.1.2.1.2 Tarea 2. Diagrama entidad relación de gestión de guías

Un diagrama o modelo entidad-relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades. Las características del modelo entidad-relación están explicadas en el apartado Diagrama entidad relación de gestión de empleados

A continuación se agregará al modelo entidad-relación existente la parte correspondiente a la gestión de guías.

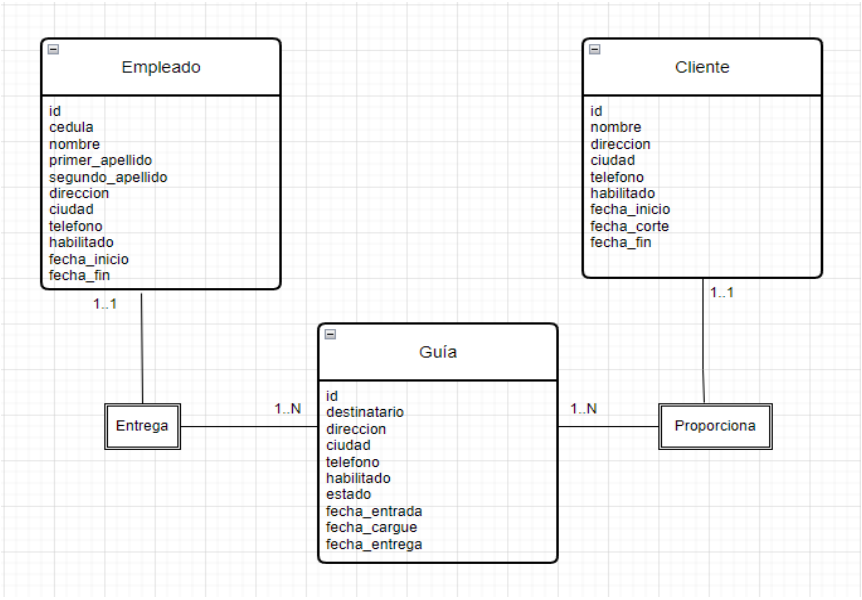


Ilustración 66. Diagrama entidad relación de gestión de guías

8.3.1.2.1.3 Tarea 3. Diagrama de clases de gestión de guías

La explicación de qué es y las características de un diagrama de clases se puede ver en el apartado Diagrama de clases de gestión de empleados

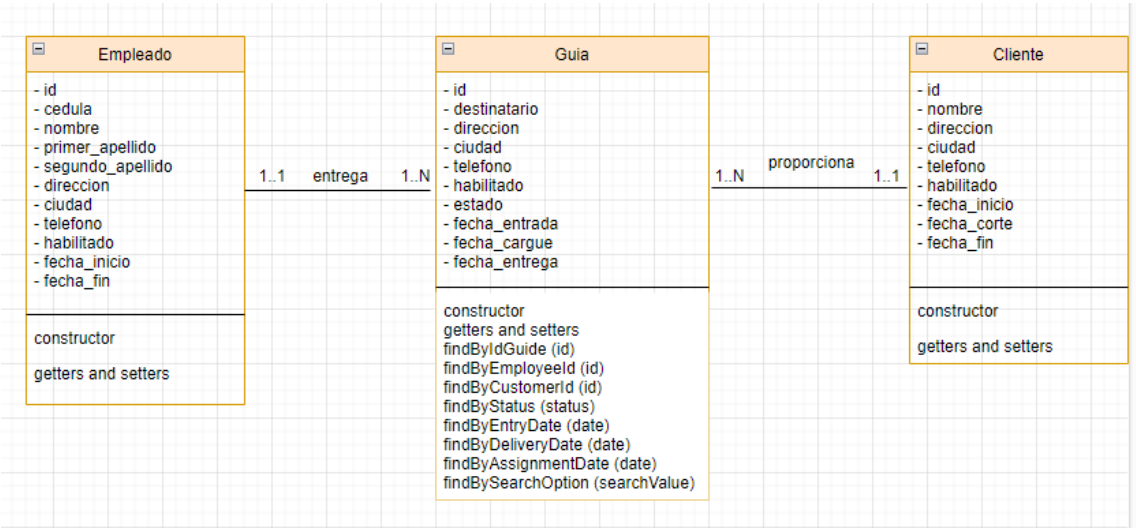


Ilustración 67. Diagrama de clases de gestión de guías

8.3.1.2.2 Historia de Usuario Diseño de gestión guía

En este apartado se va a desarrollar el diseño de gestión de guías, el cual ayuda a entender cómo desarrollaremos nuestra aplicación.

8.3.1.2.2.1 Tarea 1. Paso a tablas gestión guías

La definición y características del paso a tablas está explicado en el apartado Paso a tablas gestión empleados

Se agregará al modelo de tablas existentes la estructura correspondiente a la gestión de guías.

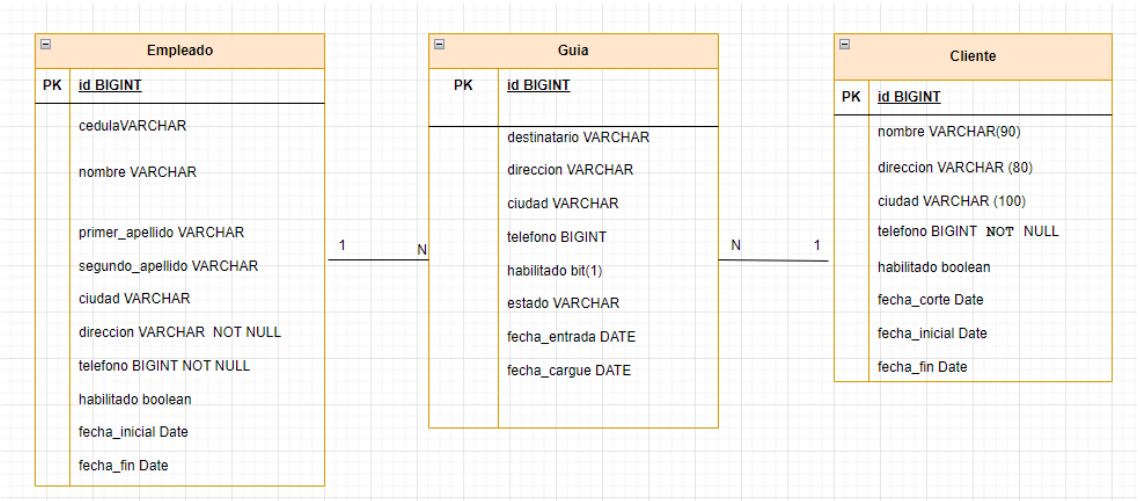


Ilustración 68. Paso a tablas gestión guías

8.3.1.2.2.2 Tarea 2. BBDD gestión guías

En cuanto al modelo de datos en este sprint corresponde en agregar las tabla guía al modelo de datos.

```
CREATE TABLE `empleado` (
 `id` BIGINT NOT NULL AUTO_INCREMENT,
 `cedula` BIGINT NOT NULL,
 `nombre` varchar(255) NOT NULL,
 `primer_apellido` varchar(255) NOT NULL,
 `segundo_apellido` varchar(255) DEFAULT NULL,
 `direccion` varchar(255) NOT NULL,
 `ciudad` varchar(100) DEFAULT NULL,
```

```
  `telefono` BIGINT NOT NULL,
  `habilitado` bit(1) NOT NULL DEFAULT b'1',
  `fecha_inicio` date DEFAULT NULL,
  `fecha_fin` date DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

CREATE TABLE `cliente` (
  `id` BIGINT NOT NULL AUTO_INCREMENT,
  `nombre` varchar(255) NOT NULL,
  `direccion` varchar(255) NOT NULL,
  `ciudad` varchar(100) DEFAULT NULL,
  `telefono` BIGINT NOT NULL,
  `habilitado` bit(1) NOT NULL default b'1',
  `fecha_corte` INT DEFAULT NULL,
  `fecha_inicio` date DEFAULT NULL,
  `fecha_fin` date DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

CREATE TABLE `guia` (
  `id_guia` BIGINT NOT NULL,
  `destinatario` varchar(255) DEFAULT NULL,
  `direccion` varchar(255) DEFAULT NULL,
  `ciudad` varchar(100) DEFAULT NULL,
  `telefono` BIGINT DEFAULT NULL,
  `habilitado` bit(1) NOT NULL default b'1',
  `estado` varchar(100) DEFAULT NULL,
  `fecha_entrada` date DEFAULT NULL,
  `fecha_cargue` date DEFAULT NULL,
```

```
  `fecha_entrega` date DEFAULT NULL,
  `id_cliente` BIGINT DEFAULT NULL,
  `id_empleado` BIGINT DEFAULT NULL,
  PRIMARY KEY (`id_guia`),
  CONSTRAINT fk_cliente FOREIGN KEY (id_cliente) REFERENCES cliente (id),
  CONSTRAINT fk_empleado FOREIGN KEY (id_empleado) REFERENCES empleado (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

8.3.1.2.2.3 Tarea 3. Diagrama secuencia de gestión guías

La explicación de qué es y las características de un diagrama de secuencia se puede ver en el apartado Diagrama de secuencia de gestión empleado

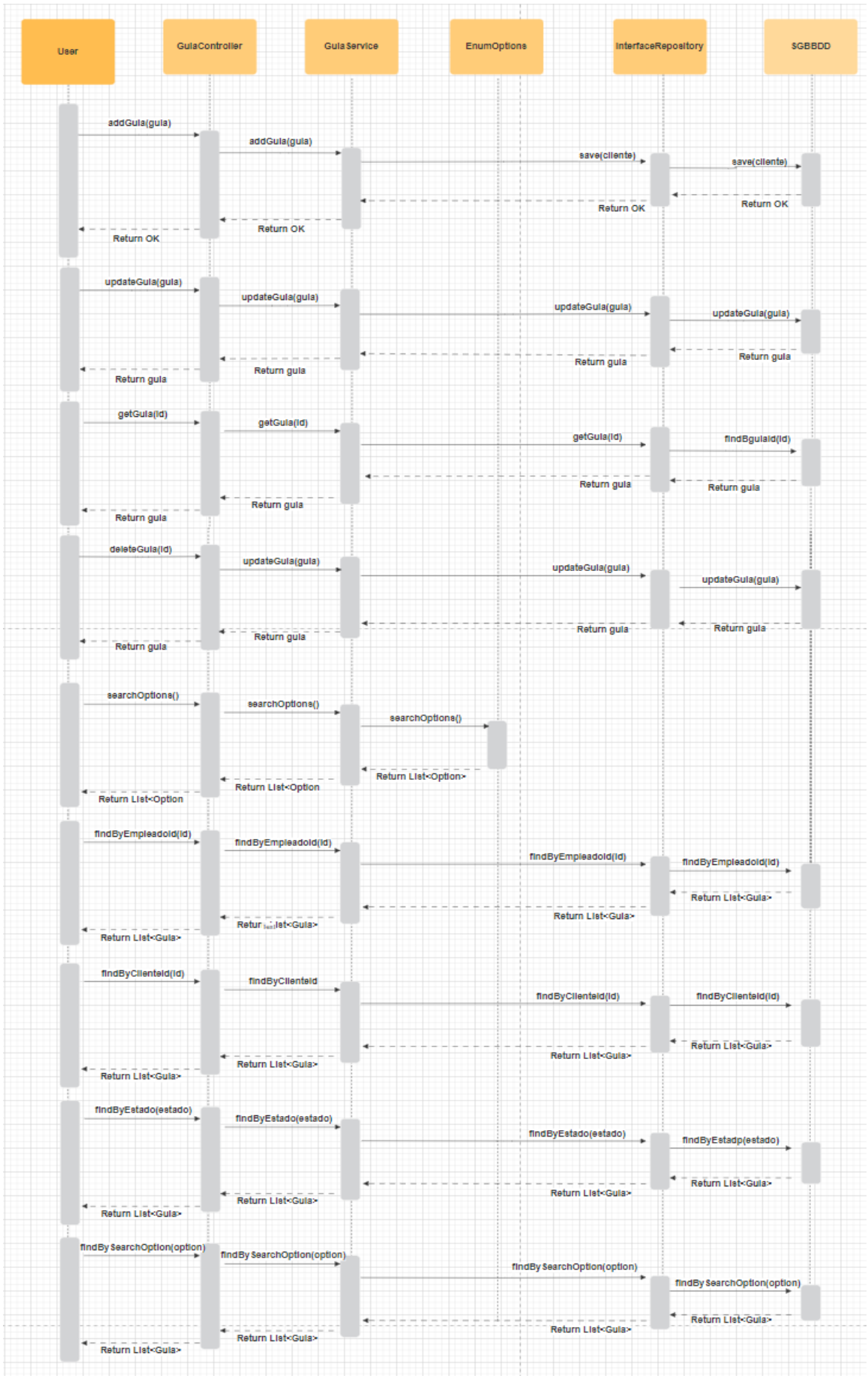

User
GuiaController
GuiaService
EnumOptions
InterfaceRepository
SGBBDD
addGuia(guia)
addGuia(guia)
save(cliente)
save(cliente)
Return OK
Return OK
Return OK
Return OK
updateGuia(guia)
updateGuia(guia)
updateGuia(guia)
updateGuia(guia)
Return guia
Return guia
Return guia
Return guia
getGuia(id)
getGuia(id)
getGuia(id)
findBguiaId(id)
Return guia
Return guia
Return guia
Return guia
deleteGuia(id)
updateGuia(guia)
updateGuia(guia)
updateGuia(guia)
Return guia
Return guia
Return guia
Return guia
searchOptions()
searchOptions()
searchOptions()
Return List<Option>
Return List<Option>
Return List<Option
findByEmpleadoId(id)
findByEmpleadoId(id)
findByEmpleadoId(id)
findByEmpleadoId(id)
Return List<Guia>
Return List<Guia>
Return List<Guia>
Return List<Guia>
findByClienteId(id)
findByClienteId
findByClienteId(id)
findByClienteId(id)
Return List<Guia>
Return List<Guia>
Return List<Guia>
Return List<Guia>
findByEstado(estado)
findByEstado(estado)
findByEstado(estado)
findByEstadp(estado)
Return List<Guia>
Return List<Guia>
Return List<Guia>
Return List<Guia>
findBy SearchOption(option)
findBy SearchOption(option)
findBy SearchOption(option)
findBy SearchOption(option)
Return List<Guia>
Return List<Guia>
Return List<Guia>
Return List<Guia>

Ilustración 69. Diagrama secuencia de gestión guías

8.3.1.2.3 Historia de Usuario Alta guía

8.3.1.2.3.1 Tarea 1. crear guía en servidor

Endpoint creado en el backend para la creación de guías.

POST/guide:

Permite la creación de una guía en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

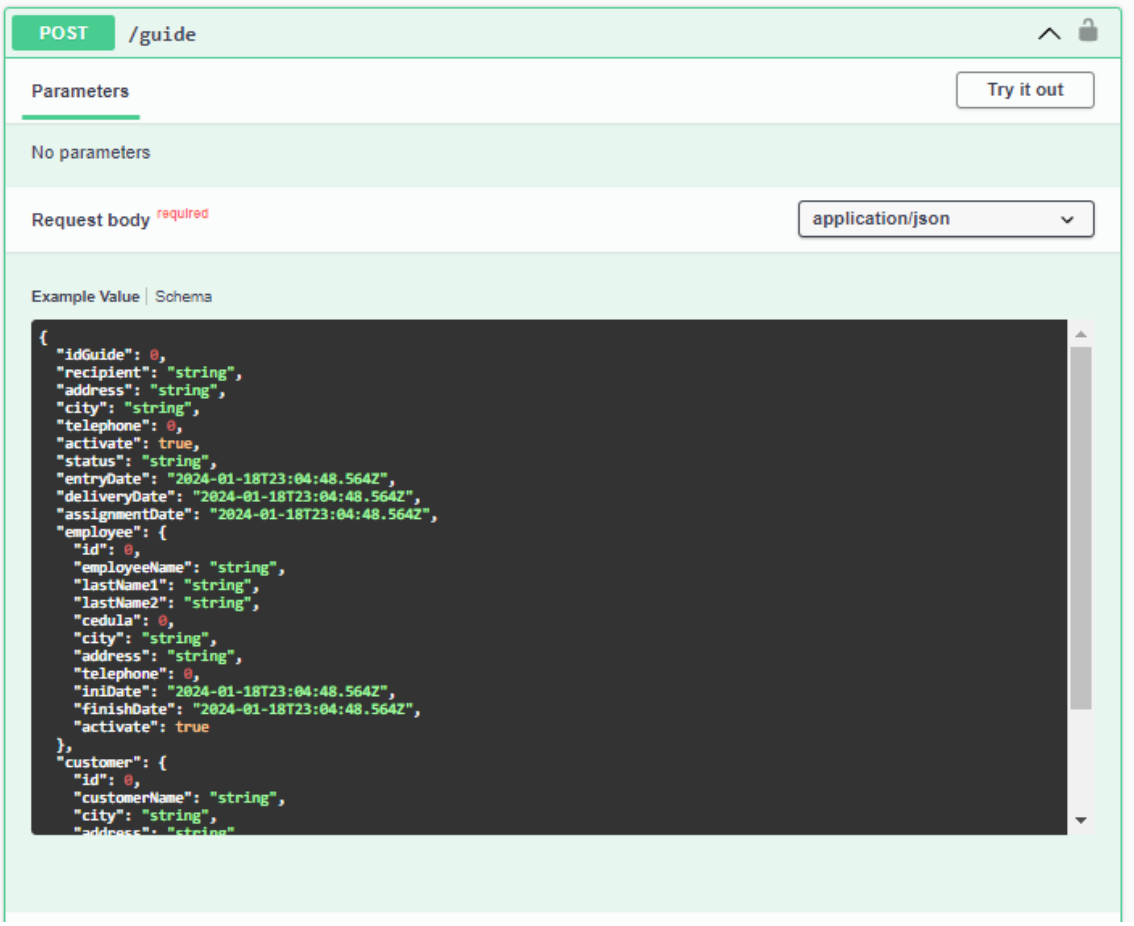


Ilustración 70. POST/guide

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para dar de alta una guía.

Al pulsar en la opción Guía del navbar, una vez se ha autenticado el usuario, éste tiene la opción de crear una nueva guía la cual representa un envío.

En la siguiente imagen vemos el formulario que se le proporciona al usuario al pulsar en el botón “Crear Guía” para introducir los datos de éste. El botón “Enviar” no estará habilitado para ser pulsado hasta que no se hayan introducido todos los datos que son obligatorios en el formulario y tampoco se habilitará si no se cumplen los requisitos de cada campo, en el caso de que no se cumplan los requisitos de cada campo se mostrará un mensaje al usuario indicándolo. También se ha agregado un botón para poner todos los campos del formulario en blanco si el usuario lo desea.

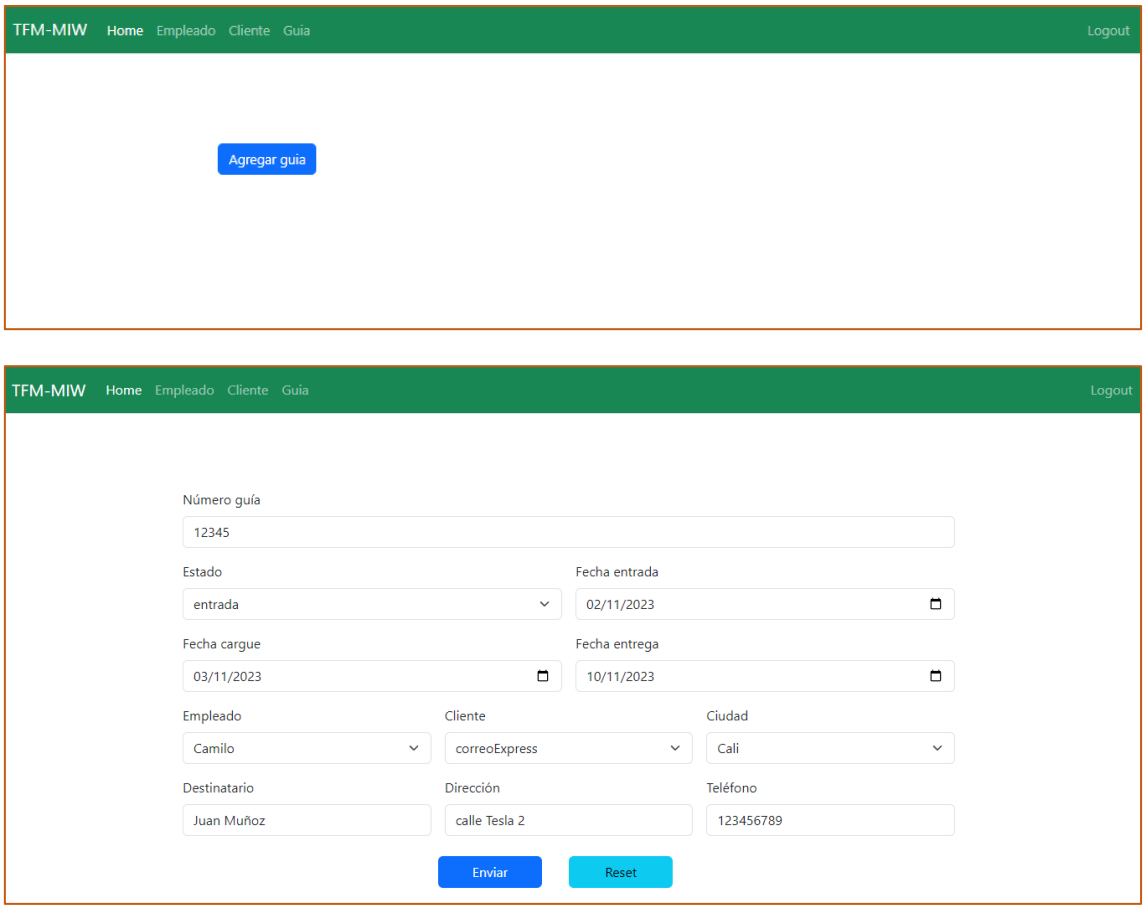


Ilustración 71. Pantallas crear guía

8.3.1.2.4 Historia de Usuario Búsqueda de guía parámetros no fecha

8.3.1.2.4.1 Tarea 1. Agregar búsqueda guía por id y estado en servidor

Endpoint creado en el backend para la realizar la búsqueda de una guía por parámetro id o por estado.

GET/customers/{id}:

Permite obtener la información de un cliente realizando la búsqueda por su id en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador de la guía y se debe enviar de forma obligatoria.

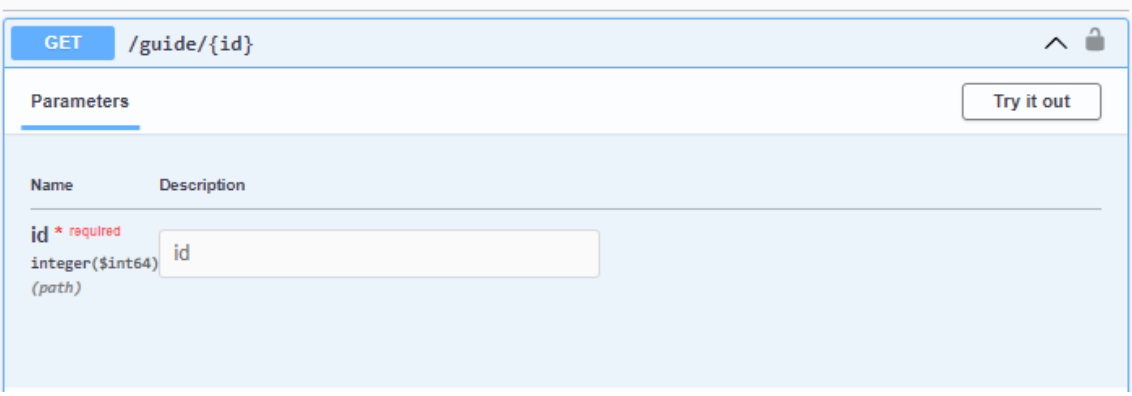


Ilustración 72. GET/customers/{id}

GET/guide/search/{searchoption}:

Permite obtener la información de un cliente realizando la búsqueda por su estado en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro searchoption que representa el estado de la guía y se debe enviar de forma obligatoria.

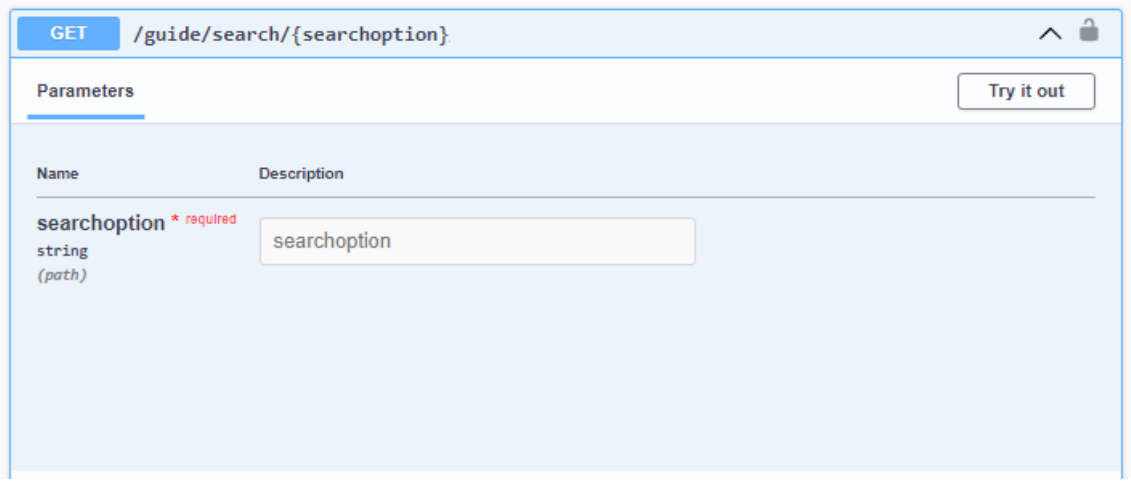


Ilustración 73. GET/guide/search/{searchoption}

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para buscar una guía por id o por estado.

Al pulsar en la opción Guía del navbar, una vez se ha autenticado el usuario, éste tiene dos opciones de búsqueda una por id de guía y la otra por estado de guía.

En la siguiente imagen vemos la interfaz de búsqueda de guía, solo se puede buscar por un tipo de parámetro a la vez y no son acumulables a la lista mostrada en la tabla; así que, si se busca por estado y hay información de guía buscada por id de guía, esa información mostrada se borrará de la tabla y se añadirá la de la nueva búsqueda realizada por el usuario o viceversa.

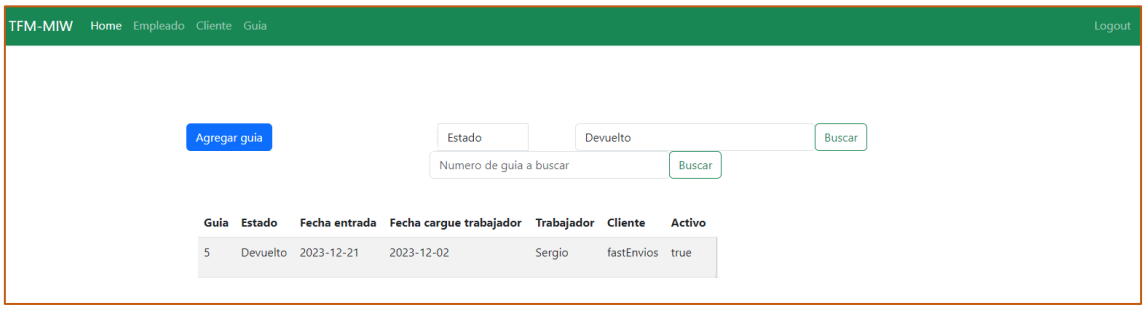


Ilustración 74. Pantalla búsqueda guía por estado

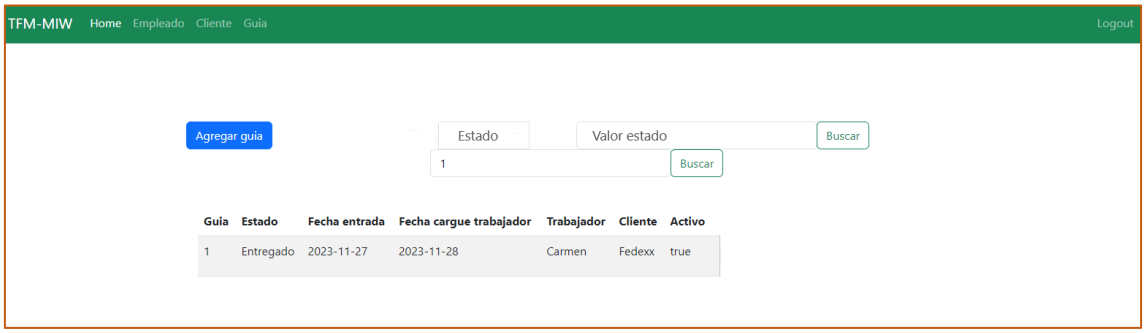


Ilustración 75. Pantalla búsqueda guía por id

Test

Cobertura: En este sprint se han realizado pruebas unitarias aunque no todas por falta de tiempo debido a que algunas tareas han tomado más tiempo de lo estimado. Una vez construido el proyecto este es el informe de cobertura de sonar.

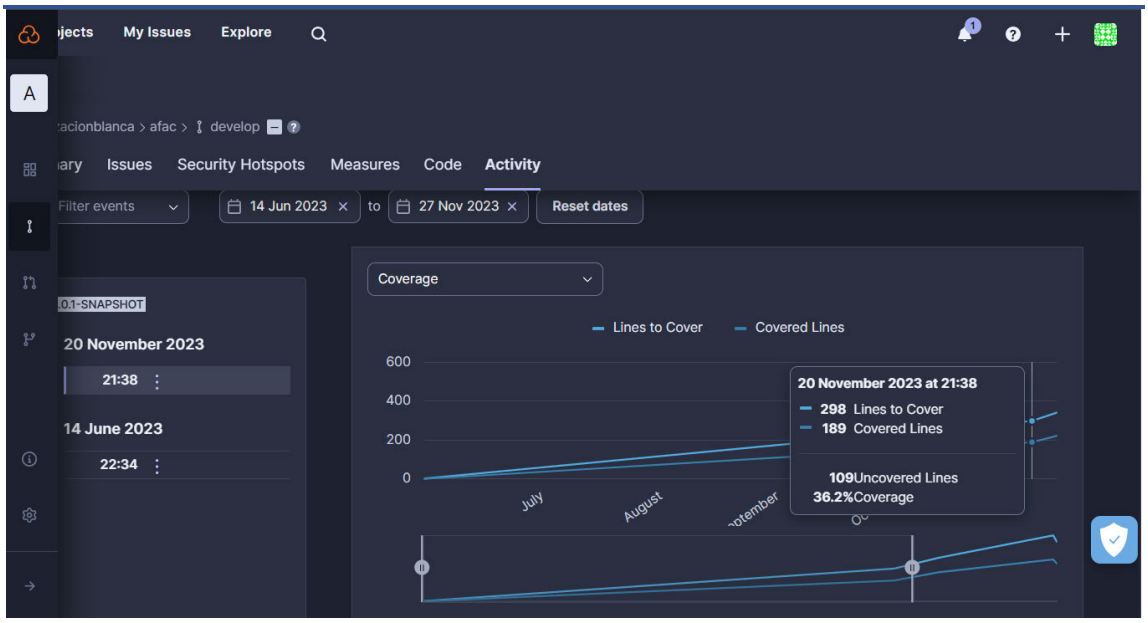


Ilustración 76. Cobertura SonarCloud sprint 2 R3

8.3.1.2.4.2 Sprint Retrospective

Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no.

En este proyecto al ser individual el alumno tomará todos los roles dentro del equipo.

Sprint Burndown

A continuación se presenta el gráfico de burdown del sprint . Como se puede apreciar, el desarrollo de este sprint se ha realizado en el tiempo estimado, en cada sprint que se termina se estima con más realidad las tareas planificadas en el sprint basándose en la experiencia adquirida en sprints anteriores.

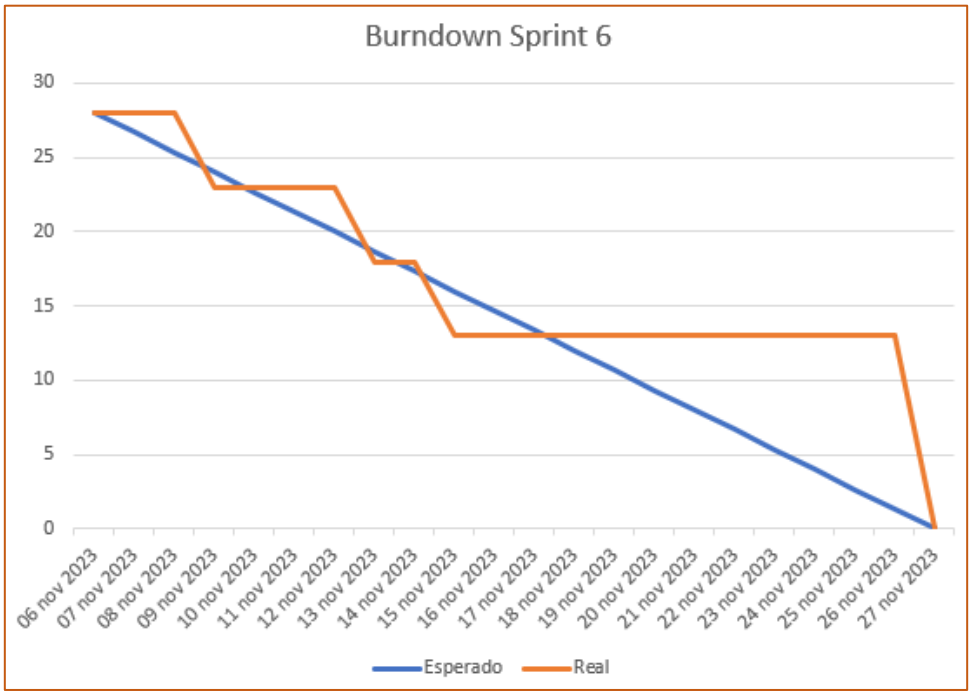


Ilustración 77. Burndown Sprint 2 R3

Para desarrollar el análisis del sprint retrospective desarrollaremos los siguientes apartados en base al desarrollo del Sprint:

- ¿Qué cosas han funcionado bien?
  Se ha conseguido terminar el Sprint con todas las tareas asignadas a éste. Se ha cubierto la cubertura que implica el desarrollo de la gestión de guías para no seguir aumentando la deuda técnica.
- ¿Cuáles hay que mejorar?

  Se sigue manteniendo la deuda técnica con respecto a cod smells y cobertura, aunque se ha abordado esta deuda técnica no se ha conseguido una cobertura del 80% que es lo aconsejable; por lo tanto, se está trabajando en que no aumente teniéndolo en cuenta en los desarrollos que se están llevando a cabo.

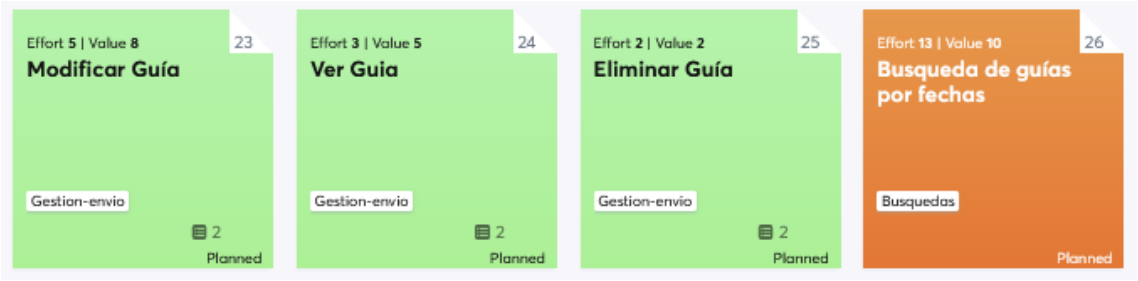
- ¿Qué cosas quiere probar hacer en la siguiente iteración?
  Realizar la cobertura del nuevo código a desarrollar.
- ¿Qué ha aprendido?
  Se ha van reforzando los conocimientos en desarrollo front-end siendo ahora fluido el desarrollo en la parte cliente de la aplicación.
- Cuáles son los problemas que podrían impedirle progresar adecuadamente.
  La incorrecta estimación de las tareas a realizar puesto que el tiempo es limitado.
  Los informes de calidad de codigo y cobertura y los valores indicados para estos

por sonar cloud cada vez que se hace la construcción de la parte cliente de la aplicación en Git Hub ya que se desea alcanzar un 80% de cobertura.

8.3.1.3 Sprint 3 R3

Sprint planing: 27-11-2023 al 17-12-2023

En este sprint se van a desarrollar las historias de usuarios de búsqueda de guías. A continuación se utilizará como herramienta visual la descripción de los casos de uso y a partir de éstos se expondrán los requisitos.



A continuación se detallarán las tareas correspondientes a las historias de usuario del backlog implicadas en este sprint.

Identificador	S5T53	**Historia de usuario**	Ver guía
Nombre	Agregar opción Ver guía parte cliente		
Descripción	Se debe crear en cada elemento de la lista de guías que son parte del resultado de la búsqueda por id o estado de la historia de usuario Historia de Usuario Búsqueda de guía parámetros no fecha un botón llamado "Ver" ubicado a la derecha de cada elemento de la lista, el cual permitirá al ser pulsado ver la información completa de la información la guía. Se debe generar la lógica necesaria para solicitar la información de la guía a un servicio expuesto en el servidor de la aplicación.		
Puntos de historia	3	**Prioridad**	Alta

Identificador	S5T54	**Historia de usuario**	Modificar Guía

Nombre	Crear modificar guía en servidor		
Descripción	Agregar la lógica necesaria para modificar una guía en Spring y exponer el servicio que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	2	**Prioridad**	Media

Identificador	S5T55	**Historia de usuario**	Modificar Guía
Nombre	Agregar la opción modificar guía en la parte cliente		
Descripción	Se debe crear en cada elemento de la lista de guías del resultado de la búsqueda realizada en la historia de usuario Historia de Usuario Búsqueda de guía parámetros no fecha un botón llamado "Modificar", el cual permitirá al ser pulsado rellenar un formulario precargado con la información modificable de la guía y mostrando la información no modificable deshabilitada. Una vez rellenado el formulario con los datos que se quieren modificar de la guía, al pulsar en el botón "enviar" se debe mostrar una ventana de confirmación al usuario donde éste confirmará la modificación de la guía o la cancelará. Se debe generar la lógica necesaria para enviar a modificar la información de la guía a un servicio expuesto en el backend de la aplicación.		
Puntos de historia	3	**Prioridad**	Media

Identificador	S5T56	**Historia de usuario**	Eliminar Guía
Nombre	Crear deshabilitar guía en servidor		
Descripción	Agregar la lógica necesaria para realizar el borrado lógico de una guía en el servidor de la aplicación y exponer el servicio que consumirá la parte cliente para esta funcionalidad		
Puntos de historia	1	**Prioridad**	Baja

Identificador	S5T57	**Historia de usuario**	Eliminar Guía
Nombre	Agregar opción eliminar guía en la parte cliente		
Descripción	En Angular, se debe agregar en cada elemento de la lista de guías que son el resultado de las búsquedas implementadas en la historia de usuario Historia de Usuario Búsqueda de guía parámetros no fecha un botón llamado "Eliminar", una guía y agregar una ventana de confirmación al usuario para deshabilitar el registro o no. Generar la lógica necesaria para deshabilitar una guía a través del servicio correspondiente expuesto en el servidor para esta funcionalidad.		
Puntos de historia	1	**Prioridad**	Baja

Identificador	S5T58	**Historia de usuario**	Búsqueda de guías por fechas
Nombre	Agregar select option para elegir el tipo de búsqueda en la vista		
Descripción	Agregar un select option para elegir el tipo de búsqueda, este select option reemplazará la búsqueda por estado implementada en la tarea Historia de Usuario Búsqueda de guía parámetros no fecha y que se cambie el input type de string a date según elija el usuario la búsqueda que quiere realizar. Se deben recuperar del servidor el tipo de opciones de búsqueda disponibles en dicho select y las opciones serán: Estado, Fecha entrada (fecha en que el cliente proporciona a la empresa el envío para entregar), fecha cargue (fecha en que se asignó dicha guía al empleado que entregará el envío) y fecha entrega de la guía.		
Puntos de historia	8	**Prioridad**	Alta

Identificador	S5T59	**Historia de usuario**	Búsqueda de guías por fechas
Nombre	Búsqueda de guía por la opción elegida en cliente		
Descripción	Se debe generar la lógica necesaria para realizar la búsqueda de guías dependiendo de la opción elegida por el usuario en frontend implementada en la tarea S5T58		

Puntos de historia	2	Prioridad	Alta

Identificador	S5T60	Historia de usuario	Búsqueda de guías por fechas
Nombre	Búsqueda de guía en servidor por la opción recibida		
Descripción	Se debe generar la lógica necesaria para realizar la búsqueda de guías dependiendo de la opción elegida por el usuario en el backend. Se expondrá un único servicio que permitirá hacer la búsqueda de las guías según el parámetro de opción de búsqueda recibido.		
Puntos de historia	3	Prioridad	Alta

8.3.1.3.1 Historia de Usuario Ver guía

Este es el resultado de la generación del código necesario tanto en FrontEnd y BackEnd para ver los datos detallados de una guía.

Al pulsar en la opción Guía del navbar, una vez se ha autenticado el usuario en el sistema, cuando se realice una búsqueda y se cargue la lista de guías se podrá hacer click en el botón “Ver” que se encuentra a la derecha de cada guía de la lista, el cual permite acceder a la vista detallada de la información de cada guía, la cual es obtenida a través del endpoint ***GET/customers/{id}*** implementado en la historia de usuario Historia de Usuario Búsqueda de guía parámetros no fecha y se muestra deshabilitada.

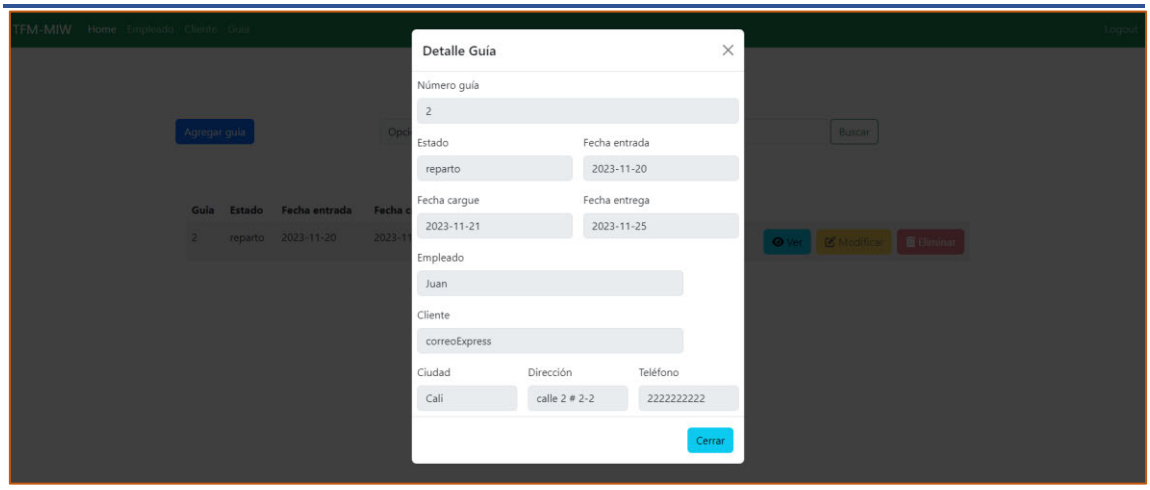


Ilustración 78. Pantallas ver guía

8.3.1.3.2 Historia de Usuario Modificar guía

Endpoint creado en el backend para la modificación de una guía.

PUT/guide/{id}:

Permite la modificación de la información de una guía en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador de la guía y se debe enviar de forma obligatoria para verificar que dicha guía existe en el sistema antes realizar el proceso de modificación.

La información del objeto enviado en el body corresponde a la información de la guía con los datos que se quieren modificar siendo obligatorio el envío de éste.

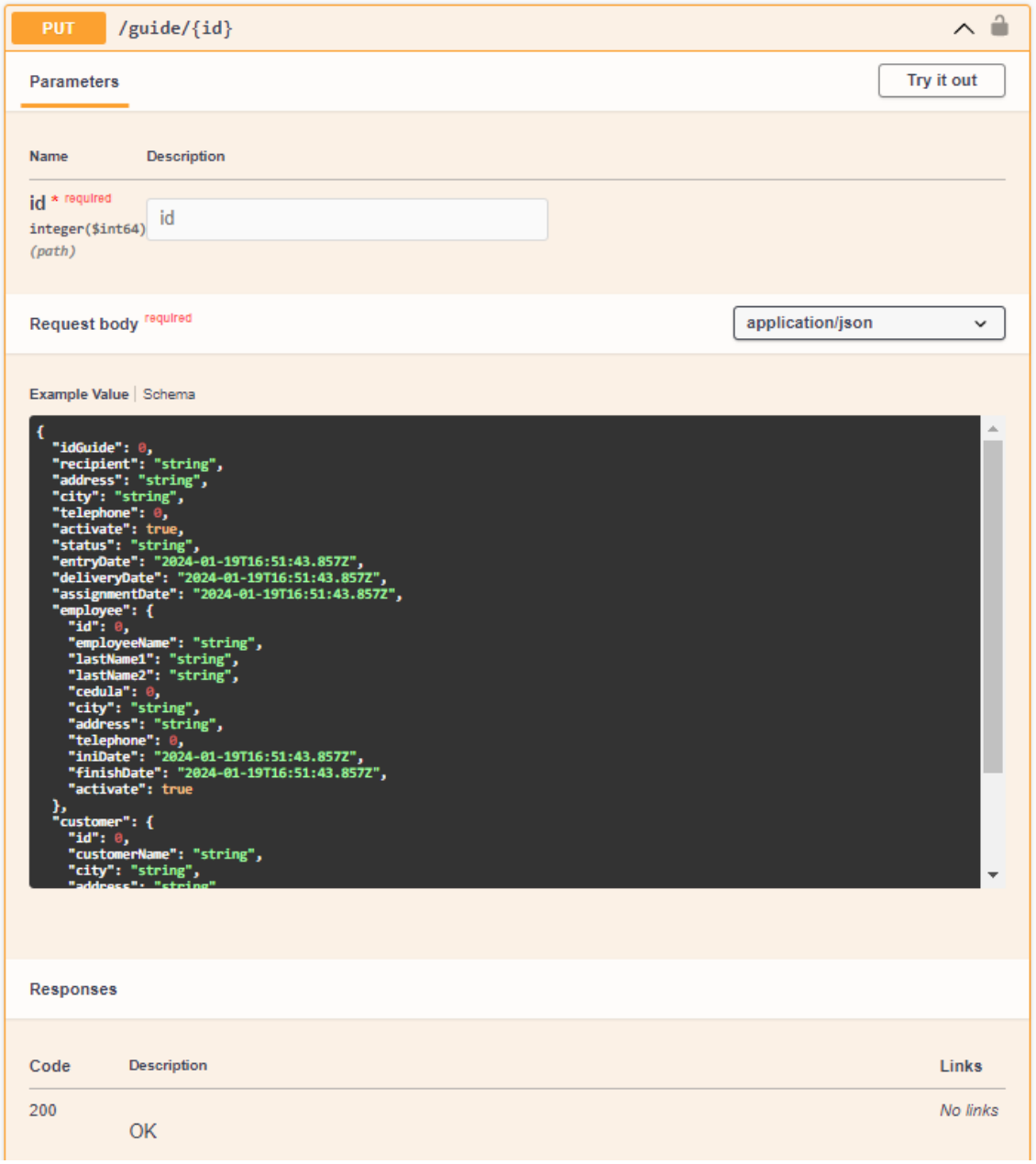
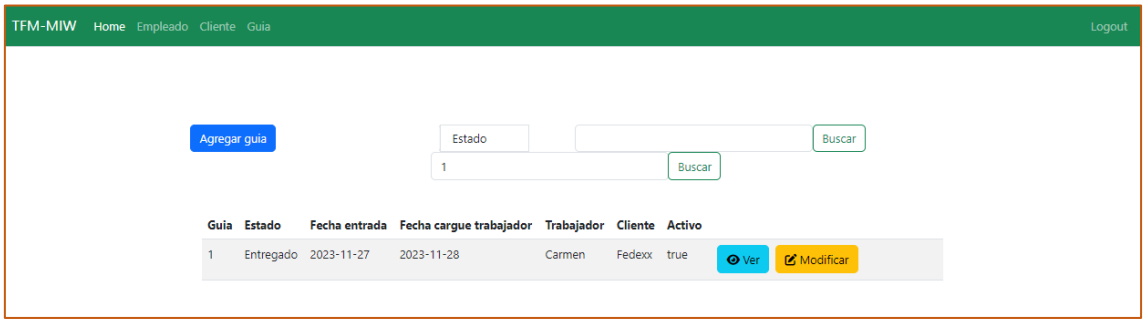


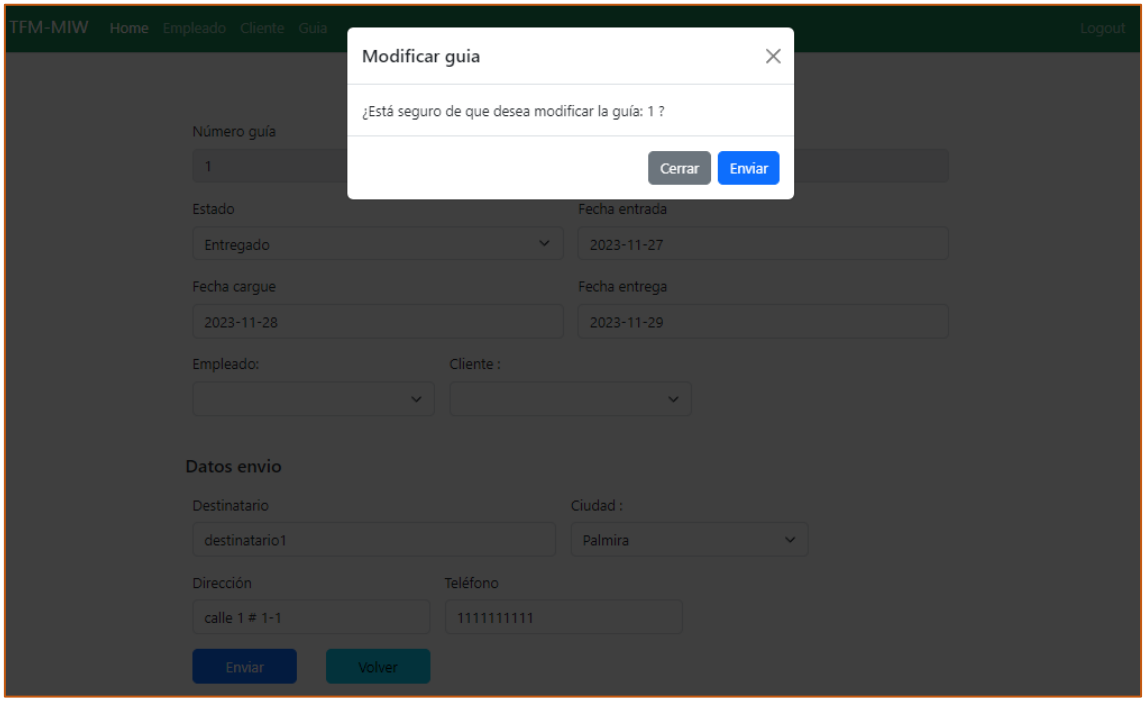
Ilustración 79. PUT/guide/{id}

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para modificar los datos de una guía de envío postal.

Al pulsar en la opción guía del navbar, una vez se ha autenticado el usuario en el sistema, el usuario tiene la opción de modificar a un cliente únicamente si se ha realizado una búsqueda con resultados, una vez visualizados los resultados de la búsqueda se tendrá la opción de pulsar el botón “Modificar” que se encuentra a la derecha de cada guía de la lista.

En la siguiente imagen vemos el formulario que se le proporciona al usuario al pulsar en el botón “Modificar” para introducir los datos de la guía que quiere modificar. La vista del formulario con la información de la guía tiene deshabilitada la información que no se puede modificar y contiene un botón de volver que lleva al usuario la vista principal de la opción de guías. También indica al usuario los campos que no cumplen los requisitos mínimos en el formulario.





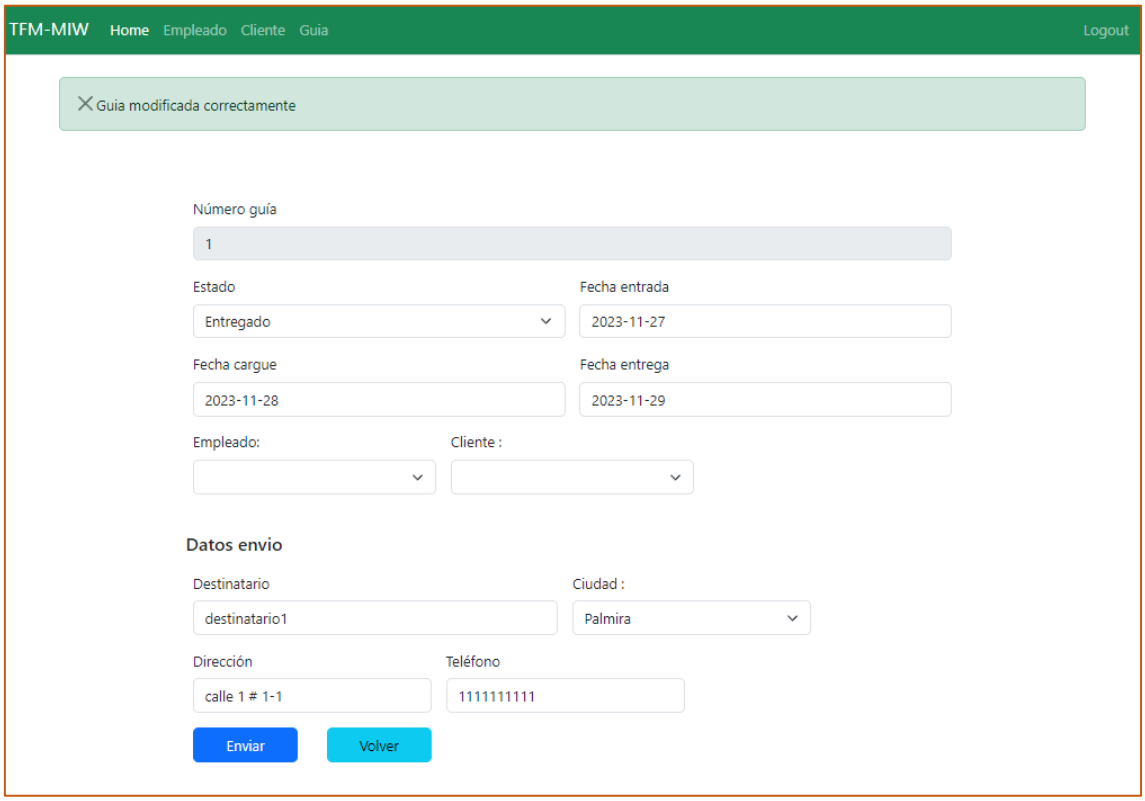


Ilustración 80. Pantallas modificar guía

8.3.1.3.3 Historia de Usuario Eliminar guía

8.3.1.3.3.1 Tarea 1. Crear deshabilitar guía en servidor

PUT/guide/disable/{id}:

Permite realizar la baja lógica de una guía en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro id es el número de identificador de la guía y se debe enviar de forma obligatoria para verificar que dicha guía existe en el sistema antes realizar el proceso de baja lógica.

A la información correspondiente al atributo actívate se le asigna el valor false y al atributo finishDate se le asigna la fecha del sistema en el momento que se accede al endpoint para posteriormente realizar el proceso de modificación de la guía.

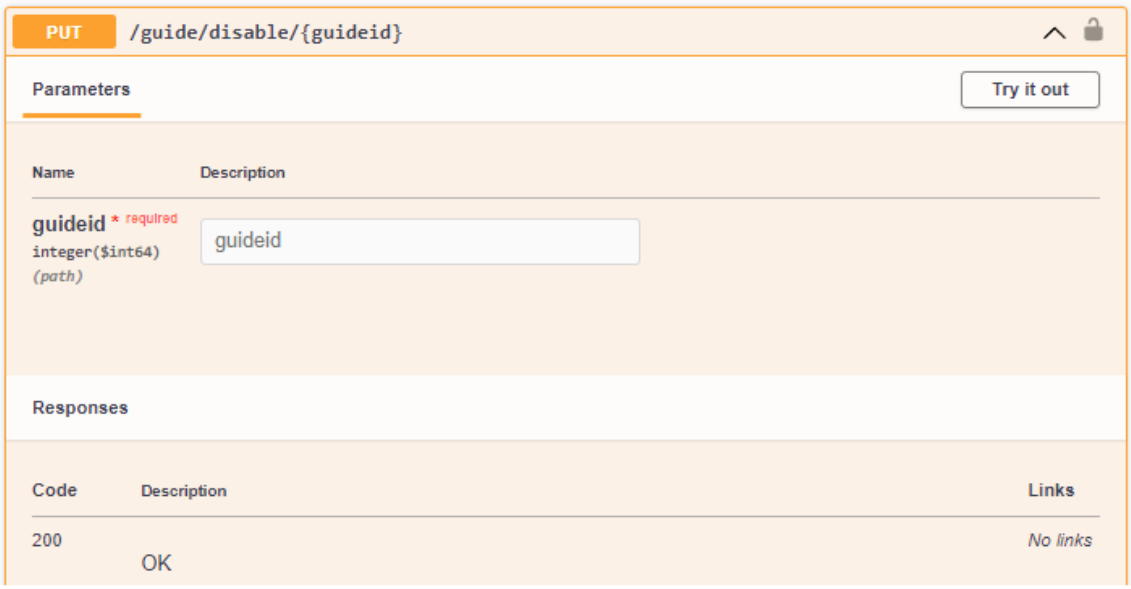


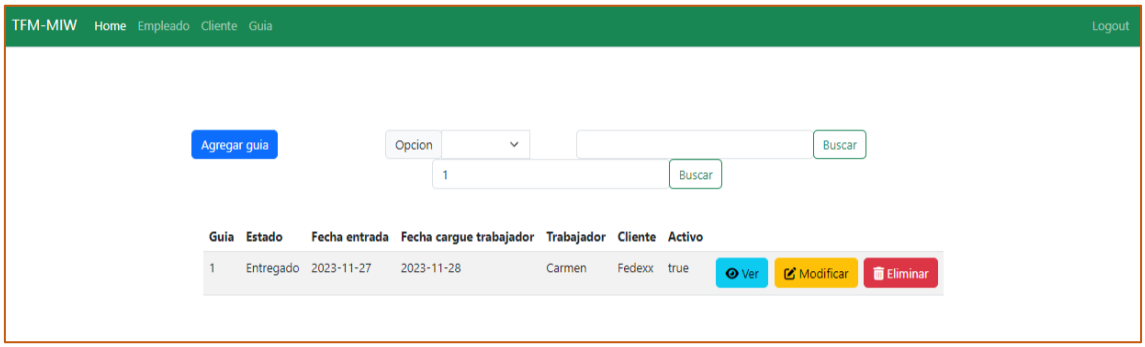
Ilustración 81. PUT/guide/disable/{id}

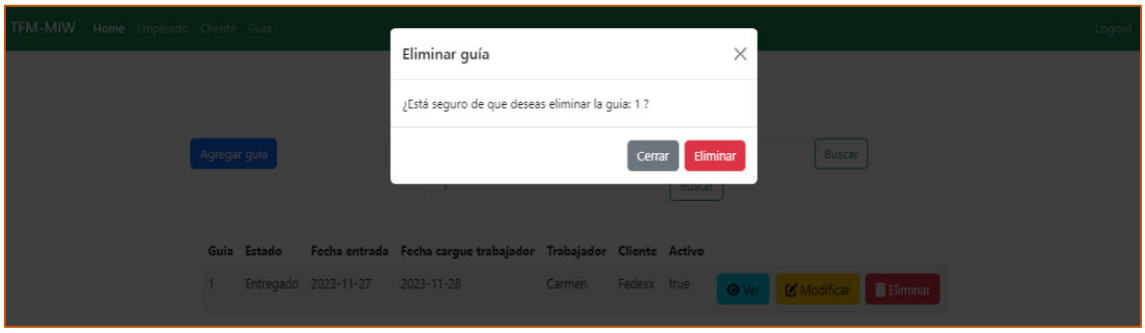
8.3.1.3.3.2 Tarea 2. Agregar opción eliminar cliente en la parte cliente

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para deshabilitar una guía de envío.

Al pulsar en la opción guía del navbar, una vez se ha autenticado el usuario en el sistema, el usuario tiene la opción de eliminar una guía únicamente si se ha realizado una búsqueda con resultados, una vez visualizados los resultados de la búsqueda se tendrá la opción de pulsar el botón “Eliminar” que se encuentra a la derecha de cada guía de la lista.

En la siguiente imagen se puede apreciar el botón eliminar a la derecha de la lista que está en la lista de resultados de búsqueda. Cuando el usuario pulse en el botón Eliminar se abre una ventana de confirmación para que el usuario acepte la eliminación lógica del registro o lo cancele.





En la siguiente imagen se puede apreciar como el botón Eliminar aparece deshabilitado si la guía de la fila de la taba de resultados de búsqueda ya está deshabilitada.

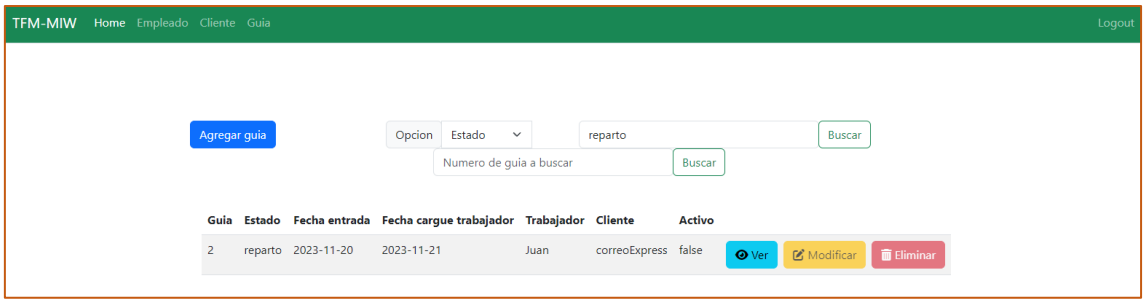
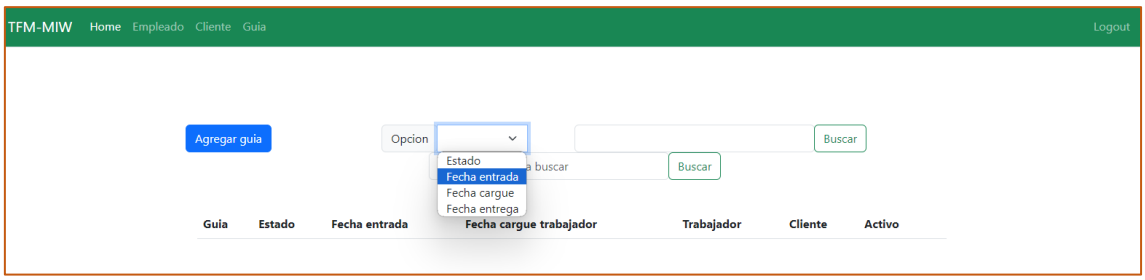


Ilustración 82. Pantallas eliminar guía

8.3.1.3.4 Historia de Usuario Búsqueda de guía por fechas

8.3.1.3.4.1 Tarea. Agregar select option para elegir el tipo de búsqueda de guía

En la siguiente imagen se puede apreciar el select de opciones agregado para la búsqueda de opciones el cual es independiente de la sección de búsqueda de guía por ID.



8.3.1.3.4.2 Tarea 3. Búsqueda de guía en servidor por la opción recibida

Endpoint creado en el backend para la realizar la búsqueda de las opciones de búsqueda que podrán ser elegidas por el usuario.

GET/guide/search/searchoption:

Permite obtener la información de las opciones de búsqueda que puede realizar un usuario en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

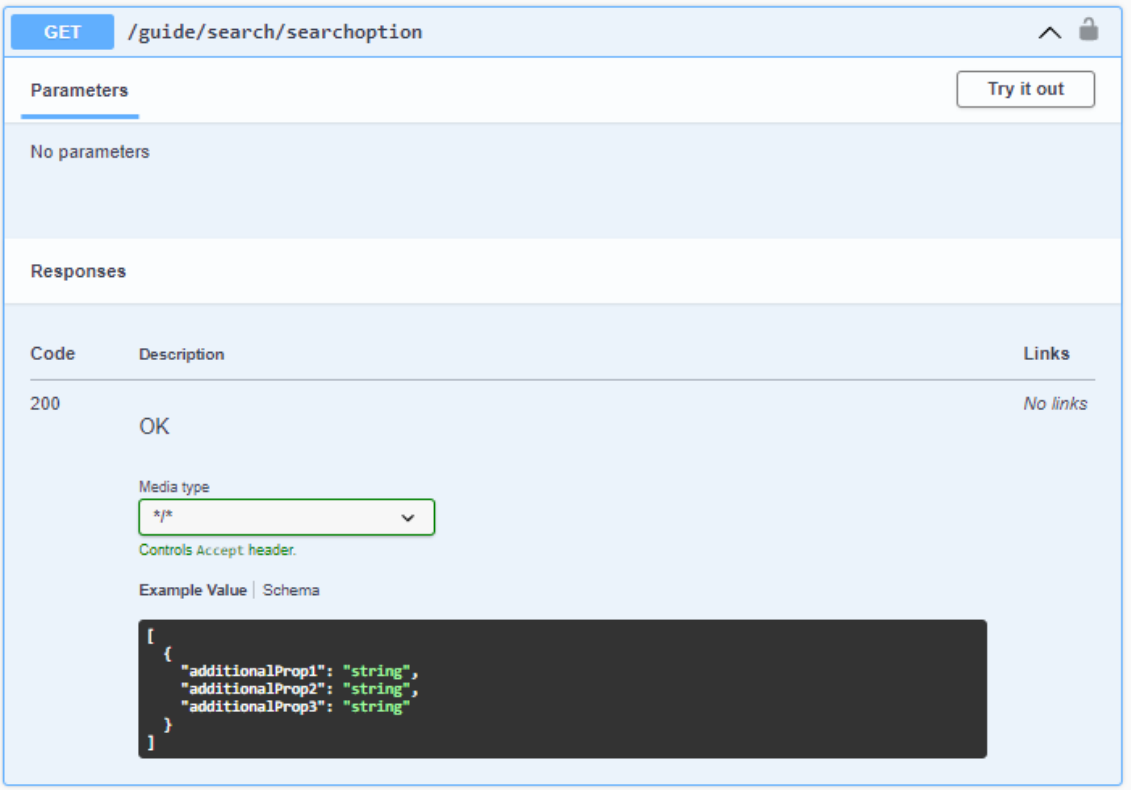


Ilustración 83. GET/guide/search/searchoption

Endpoint creado en el backend para la realizar la búsqueda de una guía por la opción elegida por el usuario.

GET/guide/search/{searchoption}/{searchvalue}:

Permite obtener la información de una guía realizando la búsqueda por su el parámetro searchvalue en el sistema. Es necesario que el usuario este autenticado y que se envíe el Authorization: `Bearer ${token}` en la petición.

El parámetro searchoption es el valor de búsqueda elegido por el usuario en el select de la vista y se debe enviar de forma obligatoria.

El parámetro searchvalue es el valor escrito por el usuario en la caja de texto que está a la derecha del select option de búsqueda y se debe enviar de forma obligatoria.

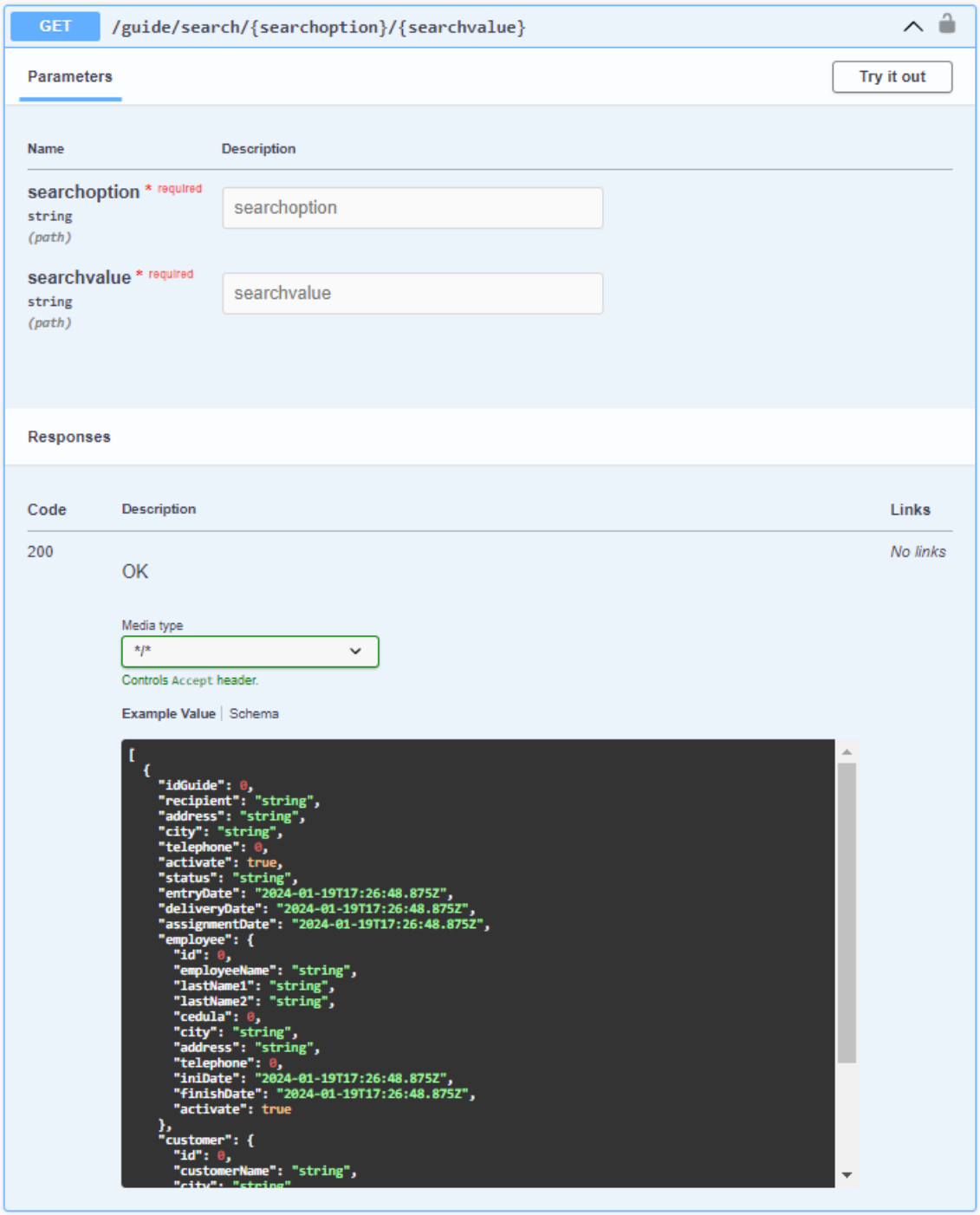
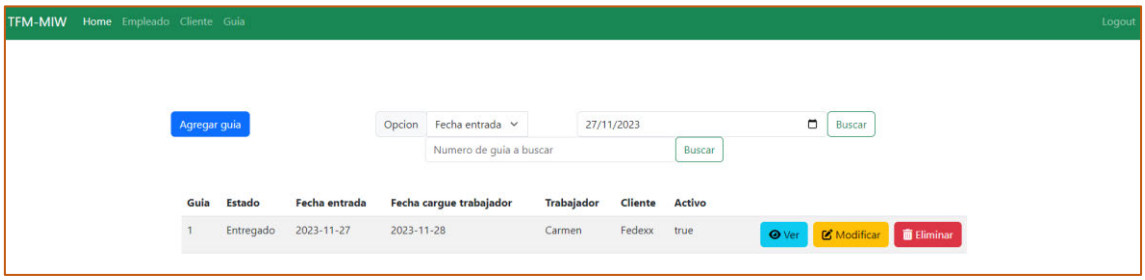


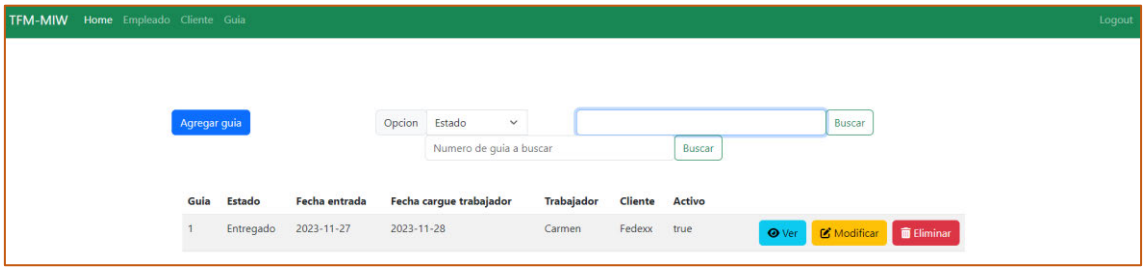
Ilustración 84. GET/guide/search/{searchoption}/{searchvalue}

Este es el resultado de la generación del código necesario tanto en FrontEnd como en BackEnd para realizar la búsqueda de guías de envío.

Al pulsar en la opción Guía del navbar, una vez se ha autenticado el usuario en el sistema, el usuario tiene la opción de realizar la búsqueda de guías eligiendo desde un select con el label “opción” el parámetro de búsqueda.



En la siguiente imagen se puede apreciar el cambio del tipo del input de Date a string en la búsqueda por opción = “Estado” . Se asume que el estado en el que esta la opción es en fecha entrada como se puede apreciar en la imagen anterior.



Al escribir la información de número de guía para realizar ese tipo de búsqueda se limpia el input de la búsqueda anterior por opción de búsqueda y la información de las guías en el listado de resultados para mostrar el resultado de la búsqueda por id de guía. Note que el registro mostrado en la siguiente imagen ya no es la guía con id = 1.

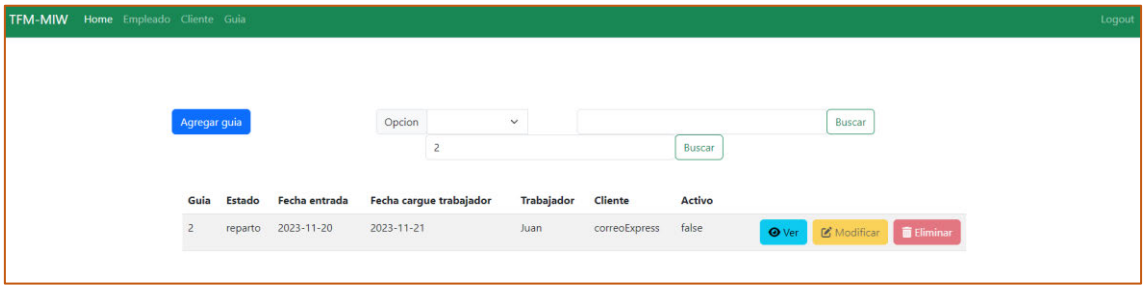


Ilustración 85. Pantallas búsqueda de guías

Integración continua

A continuación se agrega una captura de pantalla de la ejecución del action de integración continua que ejecuta los test de la aplicación y la construye en GitHub y posteriormente se ejecuta en sonar cloud el escáner de cobertura y el análisis de calidad de código del proyecto del servidor.

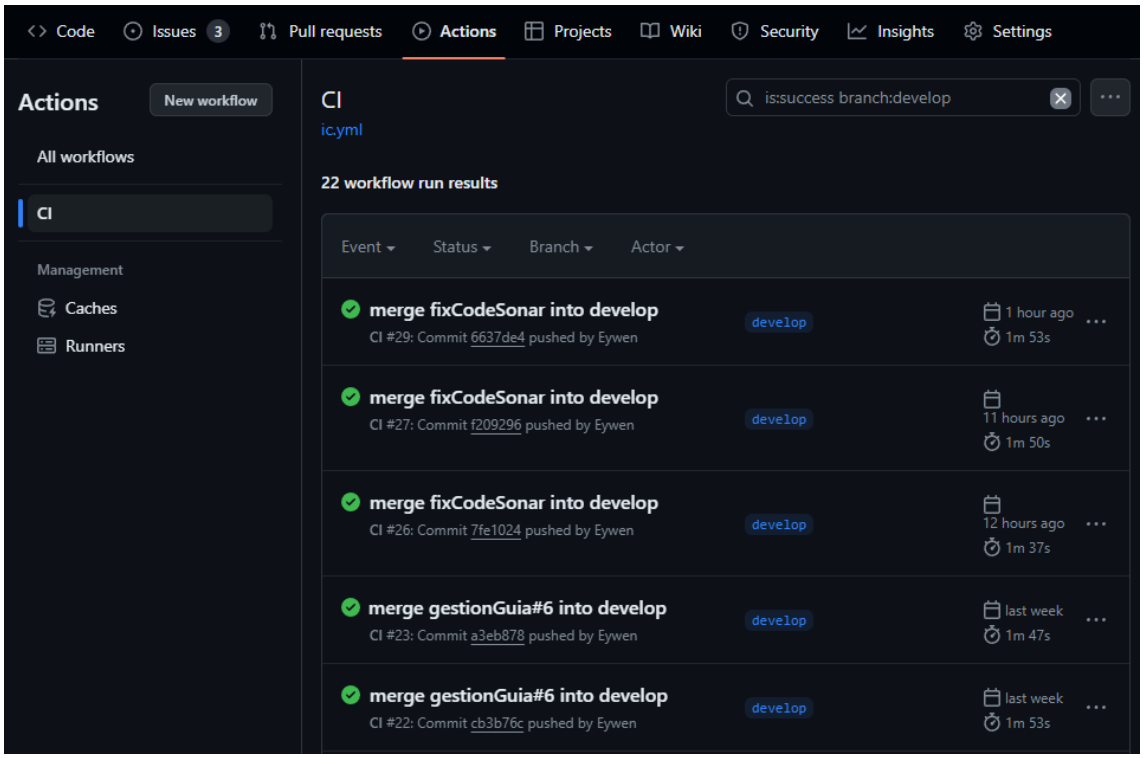


Ilustración 86. Action workflow integración continua GitHub

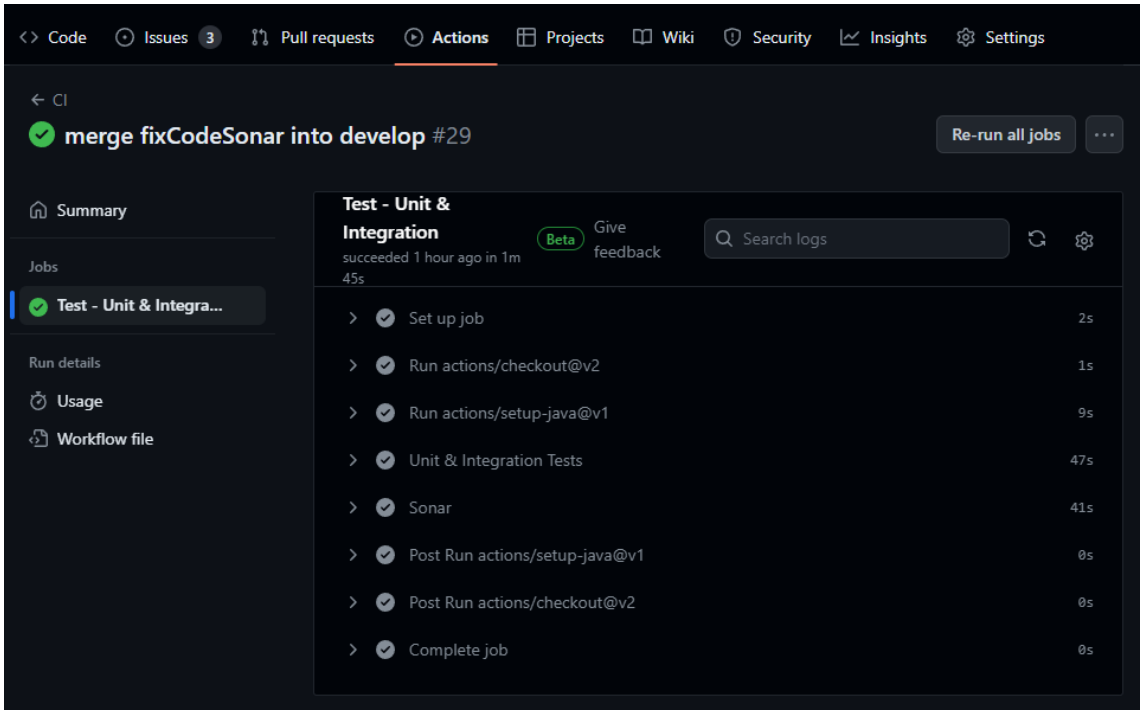


Ilustración 87. Detalle action test unit & integration GitHub

Cobertura: En este sprint se han realizado pruebas unitarias del código desarrollado y aunque la cobertura del IDE indica que se ha realizado una cobertura mucho más amplia y que se acerca al 80% deseado el informe de sonar indica valores diferentes. Se debe agregar a futuras mejoras el realizar un análisis de configuración de escáner en sonar para detectar la diferencia.

Ejecución de test del proyecto con cobertura:

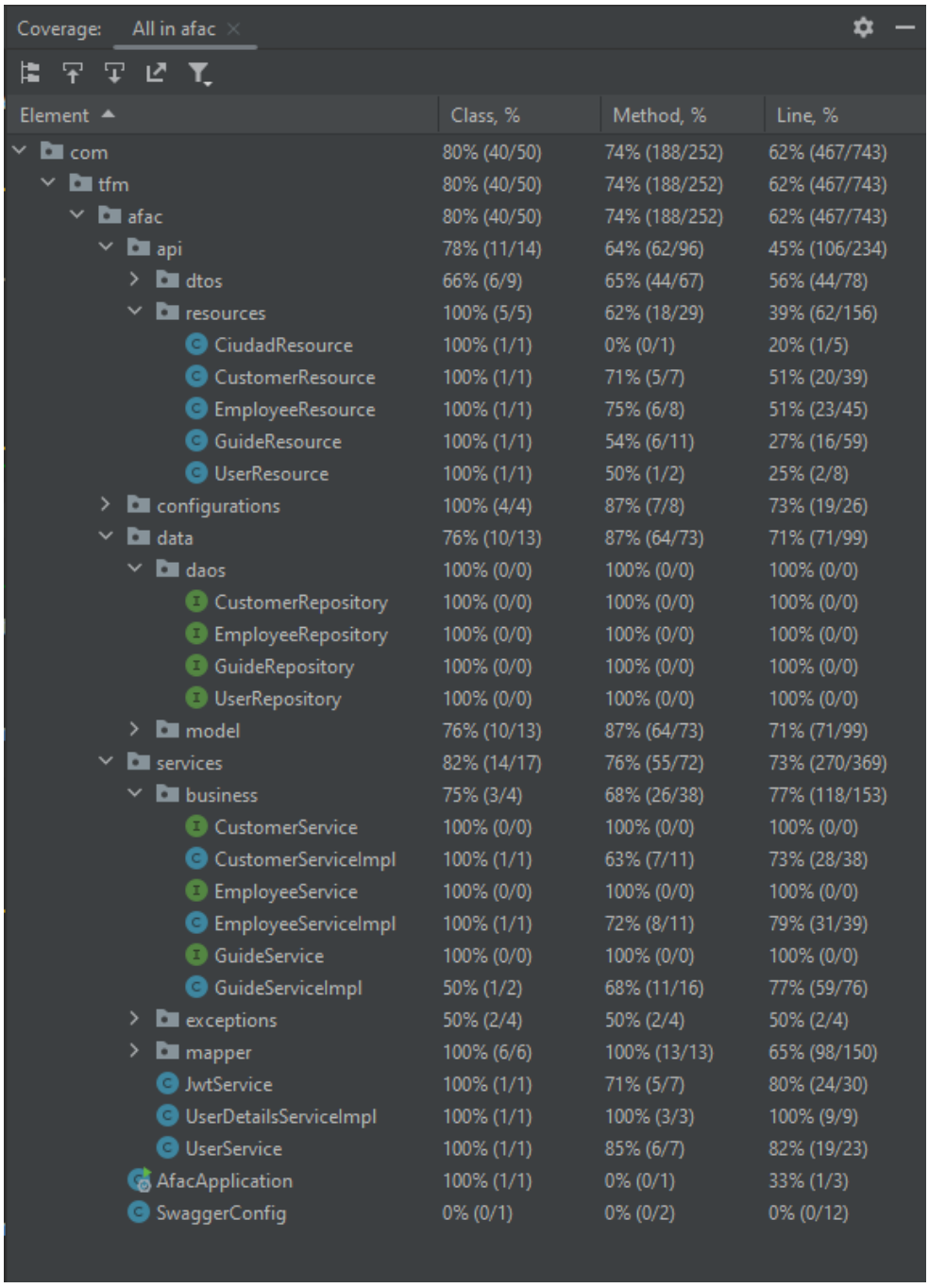


Ilustración 88. Cobertura código backend

Ejecución de scanner de sonar cloud una vez realizado satisfactoriamente el action de integración continua de git hub para evaluar la calidad del código al finalizar el desarrollo planificado para todos los sprints de esta aplicación.

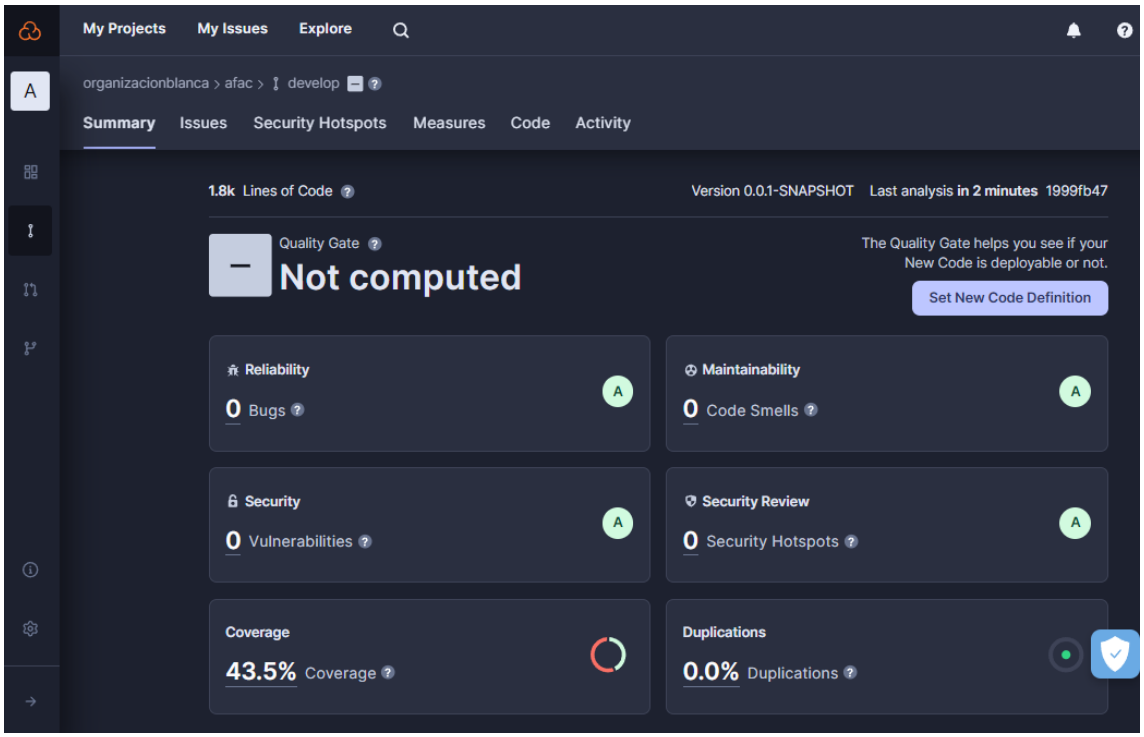


Ilustración 89. Cobertura código SonarCloud sprint 3 release 3

8.3.1.3.4.3 Sprint Retrospective

Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración, por qué está consiguiendo o no los objetivos a que se comprometió al inicio de la iteración y por qué el incremento de producto que acaba de demostrar al cliente era lo que él esperaba o no.

En este proyecto al ser individual el alumno tomará todos los roles dentro del equipo.

Sprint Burndown

A continuación se presenta el gráfico de burdown del sprint. Como se puede apreciar, el desarrollo de este sprint se ha realizado 1 día antes de lo planificado, así que se ha utilizado este tiempo para intentar aumentar la cobertura del código.

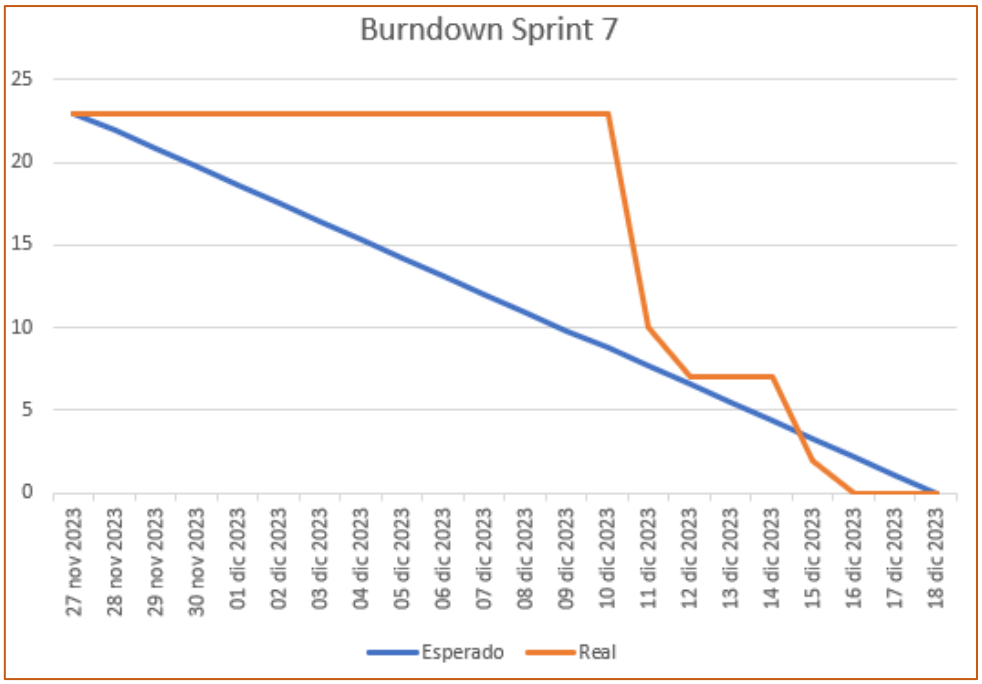


Ilustración 90. Burndown Sprint 3 R3

Para desarrollar el análisis del sprint retrospective desarrollaremos los siguientes apartados en base al desarrollo del Sprint:

- ¿Qué cosas han funcionado bien?
  Al ser este el sprint final quisiera hablar que aunque ha habido muchos momentos de bloqueo por desconocimiento en diferentes áreas finalmente con las bases adquiridas en el máster se ha podido llegar a una solución.
- ¿Cuáles hay que mejorar?

  El tiempo estimado para algunas tareas del Sprint finalmente no ha sido correcto para todas las tareas a lo largo del desarrollo aunque el sprint se ha visto en riesgo en varias ocasiones al final se pudo terminar.

- ¿Qué cosas quiere probar hacer en la siguiente iteración?
  No habrá más iteraciones pero se quedan temas fuera de este ámbito que se podrían hacer; por ejemplo, una cobertura de más del 80% del código, despliegue continuo, poner la navegación de la aplicación con https, análisis vulnerabilidades por ejemplo con OWASP, entre otras.
- ¿Qué ha aprendido?
  Se han reforzado los conocimientos adquiridos en el máster y a pesar de los inconvenientes y bloqueos se ha podido sacar adelante y solucionar los problemas que se han presentado cumpliendo con los objetivos iniciales del proyecto.

9 Conclusiones

En conclusión, en este trabajo de fin de master se han abordado los desafíos que conlleva el desarrollo de una aplicación web desde su fase inicial proponiendo una solución y enfoque que permita abordar una correcta solución basándose en los conocimientos adquiridos en este master.

Los objetivos planteados al inicio de este proyecto han sido cumplidos satisfactoriamente obteniendo como resultado una aplicación web practica y sencilla para el usuario final. El uso de la metodología Scrum y las tecnologías empleadas en el proyecto parecen haber sido adecuadas dada la finalización del mismo y la satisfacción del cliente.

Se ha conseguido desarrollar esta aplicación con una gestión de proyecto basada en metodologías ágiles, concretamente usando Scrum. El uso de Scrum para la planificación y gestión del proyecto ha permitido ver como a medida que el proyecto va avanzando surgen nuevos requerimientos y se eliminan otros. Esto ha resultado una gran ventaja para la evolución de todos los objetivos planteados inicialmente ya que ha permitido ir evaluando poco a poco si el producto que se iba obteniendo era lo que realmente el cliente esperaba y poder reajustar las tareas realizadas en cada momento.

Spring framework ha sido elegido para el desarrollo de la parte servidor creando un API Rest que expone los servicios necesarios para consumir desde el frontend. La arquitectura que se ha utilizado es una arquitectura 3 capas MVC de tal forma que se puede trabajar de manera independiente la capa de servicios, presentación y persistencia de manera limpia, cómoda y fácil.

La opción tecnológica para el desarrollo del frontend de la aplicación ha sido Angular, considerando que ha sido una elección acertada ya que ha permitido proporcionar al usuario una interfaz intuitiva y amigable.

La integración continua ha sido llevada a cabo utilizando GitHub y SonarCloud a través de workflow actions de GitHub. Lo que ha permitido tener una ejecución automática de las pruebas y construcción de la aplicación para su posterior evaluación de calidad de código.

Por último, este trabajo de fin de master se concluye cumpliendo los objetivos establecido y también se puede utilizar como una base sólida para futuras ampliaciones.

10 Posibles ampliaciones

Basándose en el producto final obtenido a lo largo de los sprints vistos en este trabajo de fin de master se pueden identificar mejoras y ampliaciones en la aplicación desarrollada.

Se pueden llevar a cabo las tareas que se quedaron pendientes:

- Cobertura del código de más del 80%
- Revisión de configuración de sonar Cloud para que coincida la cobertura de su escáner con el del IDE de desarrollo del servidor.
- Terminar de configurar el despliegue continuo que quedo incompleto por nueva elección de hosting lo cual implica un nuevo análisis de elección de hosting.

Como ampliaciones se pueden considerar llevar a cabo las indicadas en el inception deck y algunas más detectada durante el proceso de desarrollo de la aplicación hasta el momento. Las cuales son:

- En la creación de guías permitir leer el código de la guía con una pistola laser y así ofrecer al usuario la opción de no tener que digitar el número del código de barras de la guía.
- Leer desde un documento Excel el número de guía, empleado, fecha cargue y ciudad para la creación de guías.
- Genera reportes de entregas a los clientes para la facturación de los servicios prestados por la empresa.
- Generar documento pdf con los datos obtenidos en las búsquedas.
- Aplicación con https.
- Análisis de vulnerabilidades por ejemplo con OWASP una vez se haya conseguido llevar a un entorno de producción la aplicación.

11 Tabla de ilustraciones

12 Bibliografía

[1] Metodología de desarrollo. Selecting a development approach. https://web.archive.org/web/20190102182947/https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf ultimo acceso: abril de 2023

[2] Metodología de desarrollo. Velásquez Restrepo, S., Vahos-Montoya, J., Gómez Adasme M., Pino Martínez, A., Restrepo Zapata, E. y Londoño Marín, S. *Una revisión comparativa de la literatura acerca de metodologías tradicionales y modernas de desarrollo de software*. *Revista Cintex,Vol24(2)* https://revistas.pascualbravo.edu.co/index.php/cintex/article/view/334/312 ultimo acceso abril de 2023

[3] Metodología de desarrollo. Sellares, A. *Rational Unified Process (RUP).* https://ima.udg.edu/~sellares/einf-es2/present1011/metodopesadesrup.pdf ultimo acceso abril de 2023

[4] Metodología de desarrollo. *Qué es SCRUM*. https://proyectosagiles.org/que-es-scrum/ ultimo acceso abril de 2023

[5] Inception deck . Alonso, Vega A. *Recorrido por las 10 dinámicas de Agile Inception.* https://adrianalonso.es/project-management/recorrido-10-dinamicas-de-agile-inception/ ultimo acceso abril de 2023

[6] Inception deck. *The Agile Inception Deck*. https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/ ultimo acceso abril de 2023

[7] Road map. Yagüe, A. *Roadmapping*. https://prezi.com/s4leygyw8g-r/roadmapping/ ultimo acceso abril de 2023

[8] Businessmap. *¿Cómo Estructurar el Trabajo con Temas, Iniciativas, Tareas y Épicas en Agile?* https://kanbanize.com/es/agiles/metodologia-agile/epicas-temas-iniciativas-tareas ultimo acceso abril de 2023

[9] Pahíno, R. *¿Qué son Spring framework y Spring Boot?* https://www.campusmvp.es/recursos/post/que-son-spring-framework-y-spring-boot-tu-primer-programa-java-con-este-framework.aspx ultimo acceso abril de 2023

[10] Spring framework. *Introduction to the Spring Framework.* https://docs.spring.io/spring-framework/docs/4.3.x/spring-framework-reference/html/overview.html ultimo acceso abril de 2023

[11] Angular framework. *Introducción a los conceptos de Angular*. https://docs.angular.lat/guide/architecture ultimo acceso abril de 2023

[12] Documentación de GitHub. *GitHub Actions.* https://docs.github.com/es/actions/learn-github-actions/understanding-github-actions ultimo acceso abril de 2023

[13] Documentación SonarCloud. *SonarCloud Documentation*. https://docs.sonarsource.com/sonarcloud/ ultimo acceso diciembre 2023