
Grado en Ingeniería Informática

Trabajo de Final de Grado

# Simulador de Medidas para Smart Cities

*Autor:*
Adrián Adell Cantavella

*Supervisor:*
José Luis Martínez Pérez
*Tutor académico:*
Cristina Campos Sancho

Fecha de lectura: 24 de Abril de 2024
Curso académico 2023/2024

## Resumen

Esta memoria detalla el proyecto desarrollado durante las prácticas en IoTsens, cuyo propósito era crear un simulador de medidas para los sensores de la compañía. El simulador permite analizar y validar el comportamiento de los sensores en un entorno controlado, facilitando así la mejora continua de los productos de IoTsens.

En el proyecto se emplearon tecnologías consolidadas en el sector, utilizando MariaDB y la base de datos NoSQL Cassandra para la gestión eficiente de las medidas de los sensores. Para la implementación integral de la aplicación web, se optó por Spring y Angular.

Por otro lado, se adoptó la metodología Scrum para la gestión del proyecto, lo que permitió una planificación flexible y adaptativa frente a la incertidumbre del alcance del simulador.

La estructura de la memoria refleja la evolución y adaptación del proyecto a lo largo del tiempo, documentando las decisiones tomadas y las lecciones aprendidas. Este enfoque proporciona una visión transparente del proceso de desarrollo y subraya la importancia de la flexibilidad y la iteración en la creación de soluciones innovadoras en el campo de las Smart Cities.



## Palabras clave

Internet de las Cosas, Ciudades Inteligentes, Simulación, Sensores, Medidas, Java, Angular

## Keywords

Internet of Things, Smart Cities, Simulation, Sensors, Measures, Java, Angular

# Índice general

# Capítulo 1

# Introducción

Este Trabajo Fin de Grado (TFG), se elabora a partir del proyecto llevado a cabo en las prácticas en empresa. Estas prácticas permiten a los estudiantes de Ingeniería Informática integrarse y desenvolverse en un entorno de trabajo real, donde se pueden aplicar las habilidades y conocimientos adquiridos durante su formación. A su vez la empresa proporcionará los medios necesarios para llevar a acabo la tarea encomendada. Como resultado de esta experiencia, se redacta el TFG, que es la memoria donde se describe el proyecto desarrollado.

En este caso, la empresa IoTsens propuso unas prácticas nominativas, vinculadas al itinerario de Ingeniería del Software. El objetivo principal del proyecto propuesto era implementar una aplicación web que permitiese simular escenarios de Smart City, para analizar casos de uso, rendimientos de los sistemas de captación de datos y validación de los datos enviados.

Este capítulo presenta una descripción detallada de la empresa donde se realizaron las prácticas. Además, se proporciona una definición de las características del proyecto y del producto resultante.

Por último, se definen conceptos clave para poder comprender la memoria y el porque de la importancia de este producto para la empresa.

## 1.1. Sobre Grupo Gimeno

El Grupo Gimeno es una corporación con más de un siglo y medio de trayectoria, que ofrece diversos servicios a la ciudadanía mediante sus empresas en distintos sectores, como construcción, tecnología o medio ambiente entre otros, tal y como se puede observar en la figura 1.1. Algunos de estos sectores se encuentran íntimamente vinculados con servicios públicos.

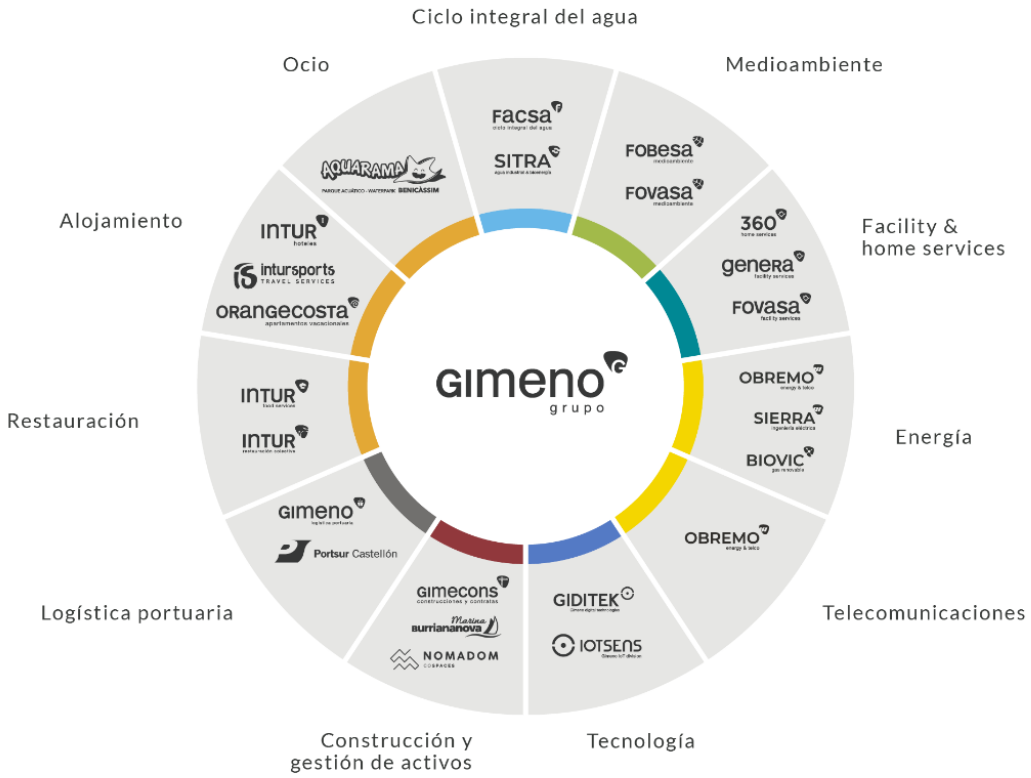



Figura 1.1: Empresas del Grupo Gimeno[7].

Dentro del grupo se encuentran, entre otras, empresas como Facsa, que se encarga del ciclo integral del agua y da servicio a más de dos millones y medio de personas; Fobesa y Fobasa, que se ocupan de la protección del medio ambiente con actividades como la limpieza, la jardinería o la gestión de ecoparques; Obremo, que se dedica a las energías renovables, el gas, la electricidad y las telecomunicaciones; Gimenocons, que realiza obras civiles y edificaciones de todo tipo; Gimeno Marítimo, que ofrece servicios de transporte internacional de mercancías; Intur, que gestiona la logística de comedores y hoteles; y Aquarama, que es un parque acuático de referencia en la costa mediterránea.

Además de las empresas anteriores, encontramos, dentro del área de tecnología, a Giditek e IoTsens[10], que merecen especial mención ya que es en ellas donde se ha desarrollado el proyecto de prácticas.

**Giditek** es una empresa clave dentro del Grupo Gimeno, lidera la transformación digital y tecnológica, impulsando la innovación a través de la adopción de nuevas tecnologías y procesos digitales. Su enfoque se centra en optimizar las operaciones y enriquecer la oferta de servicios del grupo, asegurando que se mantengan a la vanguardia de la industria.

**IoTsens**, por su parte, juega un papel crucial en el desarrollo de soluciones tecnológicas avanzadas, especializada en el desarrollo de soluciones de Internet de las Cosas (IoT), Big Data, Inteligencia Artificial, Industria 4.0, Smart Cities, Smart Water y Custom IoT. Su objetivo es alcanzar innovaciones que favorezcan la gestión eficiente de los recursos y un desarrollo sosteni-

ble. Así pues, las soluciones que ofrece están diseñadas para capturar, analizar y aprovechar los datos de los sensores que comercializan, de manera que mejoren la eficiencia, la sostenibilidad y la calidad de vida en entornos urbanos y empresariales creando un ecosistema inteligente que integra tecnología de vanguardia. Es precisamente en IoTsens, donde se ha llevado a cabo el proyecto objeto del TFG.

## 1.2. Sobre IoTsens

Como se puede ver en la figura 1.2, originalmente, IoTsens trabajaba para otras empresas dentro del Grupo Gimeno. Sin embargo, fue en 2014, después de completar varios proyectos exitosos dentro del grupo, cuando la empresa se lanzó al mercado nacional e internacional.


Figura 1.2: Creación y expansión de IoTsens[11].

IoTsens, como se ha mencionado en el apartado anterior, es una empresa que, apoyándose en la I+D, busca la optimización de los procesos, ofreciendo, entre otros servicios, soluciones integrales para el Internet de las Cosas dentro de cuatro grandes áreas:

- **City**: es una solución End-to-End que proporciona la información necesaria para la toma de decisiones en entornos urbanos.
- **Water**: es una solución integral para la gestión del agua con el fin de digitalizar las instalaciones y conseguir una mayor eficiencia.
- **Building**: es una solución para la gestión y control de edificios inteligentes con el fin de aumentar la eficiencia, la seguridad y la accesibilidad del edificio.
- **Irragation**: es una solución integral para la automatización de las zonas de riego con el fin de lograr la optimización de recursos.

En todas estas soluciones es imprescindible recabar y analizar la información relevante para la toma inteligente de decisiones. Además, para conseguir estas soluciones integrales, IoTsens ofrece una gran variedad de productos adicionales, como controladores, monitores, sensores y otros dispositivos diseñados para diferentes mercados, así como soluciones para la lectura remota de contadores inteligentes, automatizaciones, etc. Todo ello se desarrolla en distintas áreas clave como el Ciclo del agua, Conectividad, Movilidad, Software (Plataforma Iotsens) y medio ambiente.

Hoy en día, IoTsens sigue creciendo y colaborando con diversas empresas, municipios y, en general, cualquier entidad que necesite los productos y servicios que ofrecen, incluyendo proyectos a medida.

## 1.3. Qué es una Smart City

Una ciudad inteligente, o Smart City, es aquella ciudad que utiliza tecnología y soluciones digitales para mejorar la calidad de vida de sus habitantes, optimizar los recursos y servicios urbanos, y reducir el impacto ambiental [15].

En una Smart City, se implementan diversas tecnologías y dispositivos, que contienen sensores y contadores inteligentes, para recoger y procesar datos en tiempo real. Estos sensores están agrupados en dispositivos que se distribuyen por toda la ciudad. En la figura 1.3 se muestra el concepto del uso de la tecnología y la interconexión entre elementos para mejorar la calidad de entornos urbanos.

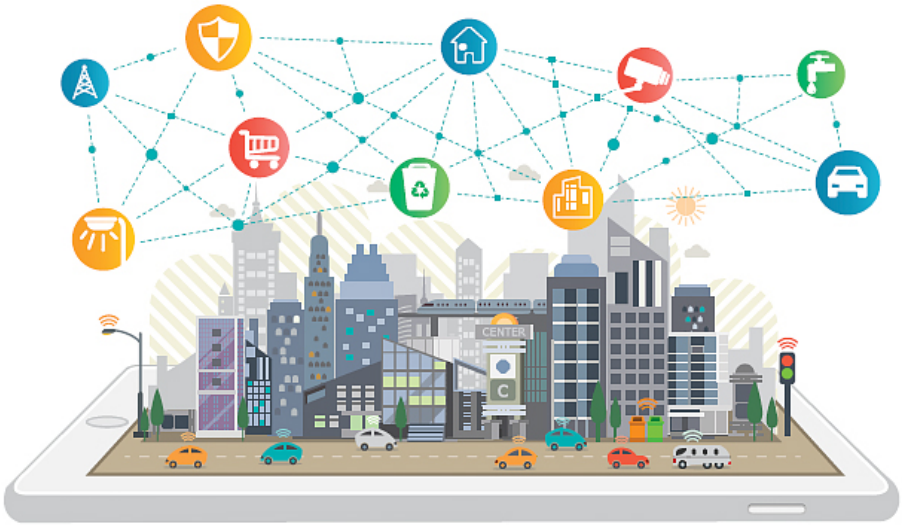



Figura 1.3: Ejemplo conceptual de una Smart City[1].

Un sensor es un elemento hardware que detecta y responde a algunas entradas de su entorno físico, como luz, calor, movimiento, presión, humedad, entre otros. Los datos recogidos pueden abarcar desde el consumo de energía y agua hasta los niveles de tráfico y contaminación.

Las empresas como IoTsens juegan un papel crucial en este ecosistema, proporcionando las soluciones y plataformas necesarias para recoger, almacenar y analizar estos datos. A través de estas plataformas, los responsables de la ciudad pueden monitorizar, gestionar y actuar eficientemente sobre los distintos aspectos de la misma.

Además, una Smart City no se limita a la gestión eficiente de los recursos, sino que también se centra en mejorar la calidad de vida de sus ciudadanos. Esto puede incluir la mejora de la movilidad urbana, la promoción de edificios y hogares inteligentes, y la creación de espacios públicos digitales.

En resumen, una Smart City es una ciudad que, gracias a la tecnología y a empresas como IoTsens, se convierte en un espacio más sostenible, eficiente y agradable para vivir.

## 1.4. Ventajas de un entorno simulado

Como se ha mencionado anteriormente, una ciudad inteligente se caracteriza por la implementación de diversas soluciones digitales, conocidas como soluciones Smart City, para mejorar la calidad de vida de sus habitantes y optimizar los recursos y servicios urbanos.

Estas soluciones se basan en dispositivos equipados con sensores que proporcionan retroalimentación continua sobre el entorno. Sin embargo, un desafío importante es que para verificar que estas soluciones funcionan correctamente en todas las situaciones posibles que los sensores podrían encontrar, tendríamos que esperar a que estas condiciones se produzcan naturalmente en el entorno donde los dispositivos están instalados. Ante la problemática presentada, resulta evidente que se debería hallar una solución. Por esta razón, se propone la creación de un simulador.

Un simulador es una herramienta esencial que permite recrear virtualmente las condiciones y escenarios que queremos estudiar, eliminando la necesidad de esperar a que ocurran en la realidad. Esto no solo ahorra tiempo y recursos, sino que también ofrece un mayor control y flexibilidad para manipular y modificar los parámetros y variables que influyen en el comportamiento de las soluciones Smart City.

En resumen, el uso de un simulador es una garantía de que las soluciones Smart City actuarán como se espera en cualquier escenario, ya que todas las posibles condiciones han sido probadas de antemano.

## 1.5. Impacto del simulador en la empresa

El desarrollo de un simulador para probar las soluciones Smart City presenta varios beneficios estratégicos para la empresa.

En primer lugar, permite realizar pruebas más eficientes y exhaustivas de las soluciones, lo que puede resultar en mejoras en la calidad y la fiabilidad de estas soluciones. Esto, a su vez, puede aumentar la satisfacción del cliente y la eficiencia operativa de la empresa.

Además, al poder probar las soluciones en una gran variedad de escenarios sin tener que esperar a que ocurran en la realidad, la empresa puede acelerar el tiempo de comercialización de sus productos y mantenerse a la vanguardia en el competitivo mercado de las Smart Cities.

Finalmente, el simulador también puede ser una valiosa herramienta de ventas y marketing. Al demostrar a los clientes potenciales como funcionarían las soluciones de la empresa en diferentes escenarios, la empresa puede aumentar su credibilidad y convencer a más clientes de los beneficios de sus productos.

## 1.6. Descripción del Producto

En los puntos siguientes se procederá a realizar una descripción general del producto a desarrollar con el fin de tener una visión global del mismo. En este caso, cabe mencionar, que se hará una especificación del producto antes que del proyecto ya que se considera que facilita la comprensión.

### 1.6.1. Objetivos

El producto a desarrollar está destinado a proporcionar al personal de IoTsens una herramienta completa y versátil que permita tomar decisiones informadas basadas en datos precisos y realistas.

Esta herramienta les permitirá consultar y analizar datos de diversos dispositivos y sensores, simular diferentes escenarios y comparar los resultados. Todo esto con la intención de facilitar la toma de decisiones basada en datos en el contexto de las implementaciones de Smart City.

Además, se pretende que el producto desarrollado, aunque sea independiente, esté integrado con la aplicación central de la empresa (Control Platform). De esta manera, el software contribuirá a mejorar y expandir sus funcionalidades, puesto que gestionará la simulación de las medidas asociadas a los sensores incluidos en dicha aplicación central.

Finalmente, la aplicación también ofrecerá una versión traducida al inglés, facilitando su utilización en ambos idiomas, inglés y español.

### 1.6.2. Alcance Funcional

En el contexto del alcance funcional del producto, el software de simulación de escenarios para implementaciones de Smart City tiene como objetivo fundamental generar datos que sean lo más realistas posible. Estos datos permitirán analizar casos de uso, evaluar el rendimiento de los sistemas de captación de datos y validar la información transmitida.

La aplicación web proporcionará al usuario la capacidad de consultar la información de los dispositivos importados desde la aplicación central de IoTsens. Ver sus sensores y las medidas que estos generen, ya que, de nuevo, cabe recalcar que los sensores son los verdaderos transmisores de la información recopilada, mientras que el dispositivo actúa como un contenedor para éstos.

Adicionalmente, el usuario podrá especificar los detalles de la simulación sobre cada uno de los sensores. Estos detalles se especificarán en la fase de análisis.

### 1.6.3. Alcance Informático

La aplicación establecerá una interacción con los sistemas de captación de datos de IoTsens, lo que permitirá simular datos de sensores sin la necesidad de instalar dispositivos físicos. Esto supone una interacción directa con los sistemas de IoTsens para la recogida, almacenamiento y análisis de datos.

Además, la aplicación se comunicará con la base de datos central de IoTsens, donde se almacena la información relativa a los dispositivos y sensores que la empresa tiene disponibles.

### 1.6.4. Alcance Organizativo

En este caso, la aplicación está diseñada para ser utilizada por IoTsens. La empresa la empleará para probar sus implementaciones y nuevas tecnologías sin la necesidad de instalar sensores físicamente. Es decir, permitirá a IoTsens simular y probar diferentes escenarios y casos de uso, analizar el rendimiento de sus sistemas y validar los datos enviados, todo ello en un entorno controlado.

Así pues, la aplicación puede ayudar a la empresa a identificar y resolver problemas antes de que las soluciones se desplieguen en el mundo real, mejorando así la eficiencia y la efectividad de sus productos. Además, brindará la oportunidad al personal de IoTsens de probar casos de uso para clientes externos, lo que podría resultar en una mayor satisfacción del cliente y un producto final más ajustado a sus necesidades.

## 1.7. Descripción del Proyecto

### 1.7.1. Objetivos

El objetivo principal de este proyecto es desarrollar un producto software que sirva para simular medidas de los sensores de los que dispone IoTsens, que están instalados en dispositivos repartidos en distintas ubicaciones.

Por otro lado, el proceso de desarrollo seguirá unos estándares de calidad que afectarán a la implementación y al resultado final del producto, ya que al mejorar la calidad del código, se mejora la calidad del producto. Estos estándares se describen en el libro de Clean Code[16] facilitado por la empresa. Se exigirán estas buenas prácticas para que la aplicación pueda ser comprensible por todos los miembros del departamento de software y conseguir que sea fácilmente mantenible y ampliable.

### 1.7.2. Alcance del Proyecto

El alcance de este proyecto engloba todas las etapas de creación del software, desde las más tempranas, como podrían ser las fases de análisis y planificación, hasta sus últimas fases, como lo serían el desarrollo y las pruebas finales. Además, este alcance abarca el despliegue de la aplicación en el entorno interno de la empresa para su uso por parte de sus desarrolladores.

## 1.8. Descripción detallada del desarrollo del proyecto

### 1.8.1. Tecnologías para el desarrollo

Para el desarrollo de este proyecto se han escogido varias tecnologías y herramientas.

Por una parte, para la interfaz de usuario, se utiliza Angular, un framework para aplicaciones web, desarrollado y mantenido por Google y que está construido en TypeScript, un lenguaje que extiende JavaScript añadiendo tipos estáticos. Esto, permite una mayor eficiencia y seguridad en el desarrollo de aplicaciones web. También se emplea PrimeNG, que es una popular biblioteca de estilos y componentes especializados para Angular.

Por otra parte, en el lado del servidor, se utiliza Spring, que es otro framework, para el desarrollo eficiente y robusto de aplicaciones en Java.

Además, se usará Cassandra, una base de datos no relacional donde se almacenará toda la información relativa a los dispositivos y sensores de la empresa. Cassandra permitirá leer y guardar medidas de manera eficiente debido a su capacidad para manejar grandes volúmenes de datos distribuidos a través de muchos servidores, lo que proporciona alta disponibilidad y resistencia a fallos.

Para el desarrollo y la gestión de la base de datos del simulador a desarrollar, se utilizará MariaDB en combinación con Flyway. MariaDB es una base de datos relacional compatible con SQL, mientras que Flyway es una herramienta que permite controlar las versiones de la base de datos. La combinación de estas dos tecnologías proporciona una solución robusta y eficiente para la gestión de la base de datos del proyecto.

Por último, mencionar que como Entorno de Desarrollo Integrado (IDE) se empleará IntelliJ Idea y Git/GitLab para el control de versiones del software en desarrollo.

En conclusión, el desarrollo de este producto software abarcará una amplia gama de tecnologías y herramientas, cada una de las cuales desempeña un papel fundamental en diferentes aspectos del proyecto. En la figura 1.4 se puede ver como interactúan las tecnologías entre sí.

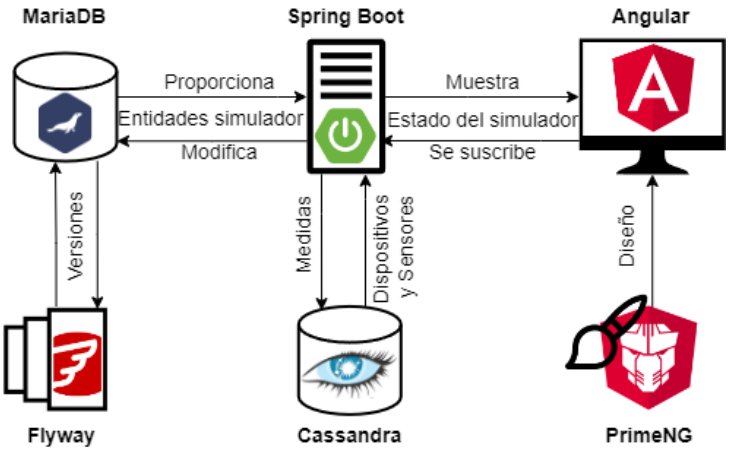



Figura 1.4: Interacción entre las tecnologías empleadas.

### 1.8.2. Estructura de la memoria del proyecto

Después de esta introducción, el informe se adentra en las diferentes fases del desarrollo del software vinculado a este proyecto. A continuación, se detallan los contenidos de cada capítulo:

- **Capítulo 2, Planificación del Proyecto**: se detalla la metodología de trabajo adoptada y su aplicación práctica en la empresa. Se introduce el primer sprint y se establece la estructura para los sprints subsiguientes.
- **Capítulo 3, Desarrollo del proyecto por sprints**: se describe el desarrollo del producto a lo largo de los sprints, incluyendo la progresión de la pila del producto. Cada sprint se examina en términos de análisis, diseño e implementación, reflejando la naturaleza no definitiva de los requisitos del simulador.
- **Capítulo 4, Resultados**: se presentan los resultados alcanzados después de todo el ciclo de desarrollo, mostrando elementos como el diseño definitivo de la base de datos, la interfaz de usuario, la pila completa del producto o las pruebas realizadas.
- **Capítulo 5, Conclusiones**: Se evalúan los objetivos logrados, los conocimientos adquiridos y aplicados y se sugieren mejoras para el futuro, junto con unas reflexiones finales.

# Capítulo 2

# Planificación del Proyecto

## 2.1. Metodología

La metodología seleccionada para el desarrollo del Trabajo de Fin de Grado es Scrum [17]. Esta elección se debe a la naturaleza incierta y compleja del proyecto. Aunque se tiene una visión general de que se quiere un simulador y las ventajas que conllevaría, el alcance exacto del proyecto y los objetivos específicos aún no están claros.

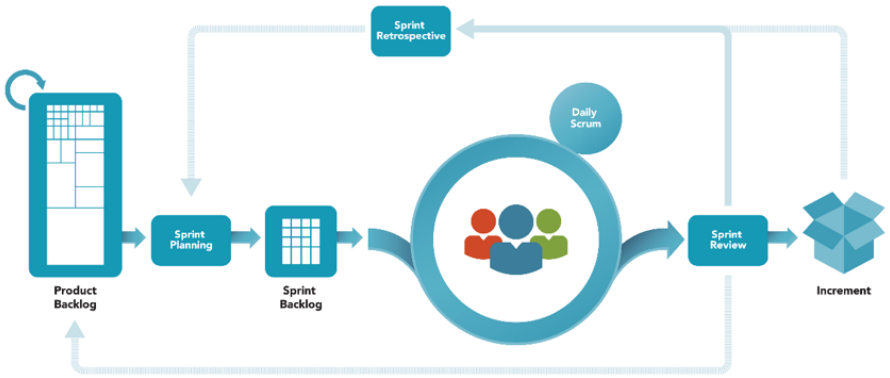



Figura 2.1: Esquema de la metodología Scrum[4].

La metodología Scrum permite abordar este tipo de proyectos en pequeñas etapas o sprints, facilitando así la gestión y el control del progreso. Su flexibilidad y capacidad para adaptarse a los cambios son especialmente útiles en este contexto. A medida que el proyecto avanza y se obtiene más información, se pueden hacer ajustes y refinamientos, lo que permite una mayor precisión en la planificación y ejecución. En la figura 2.1 se puede ver, de forma esquemática, la secuencia de esta metodología.

## 2.2. Adaptación de la Metodología en la Empresa

En la empresa, la metodología Scrum se adapta de manera que todos los días a las 9 se realiza una reunión diaria o *daily meeting*. La *daily* es una reunión corta en la que cada miembro del equipo informa sobre lo que ha hecho el día anterior, lo que va a hacer el día de hoy y si tiene algún impedimento para realizar su trabajo.

En segundo lugar, los sprints tienen una duración de dos semanas, y los requisitos se establecen mediante historias de usuario. Además, se crea, automáticamente, un tablero Kanban[5], donde cada columna representa una etapa de desarrollo del producto, viéndose claramente su estado. En el mismo tablero se muestran las tareas individuales con el trabajo a realizar, permitiendo así una gestión más cómoda e interactiva.

Después de cada sprint, se realiza una demostración con el supervisor de las prácticas para mostrar el progreso y los resultados obtenidos. Durante estas demostraciones, también se recibe feedback del supervisor, lo cual es crucial para el aprendizaje y la mejora continua. A continuación, se planifica el siguiente sprint, permitiendo ajustar el enfoque y las prioridades en función de los comentarios y los resultados de la demostración.

Es importante mencionar que las bases de datos relacionales requieren un análisis profundo al principio del desarrollo para minimizar futuras modificaciones, ya que los cambios significativos son muy costosos y pueden afectar negativamente su rendimiento y estabilidad. En este caso, al carecer de ese análisis detallado previo se necesita utilizar una herramienta que permita realizar modificaciones incrementales sobre la base de datos durante los sprints. La herramienta empleada para garantizar que el sistema de almacenamiento se ajuste correctamente a la metodología es Flyway. Esto permite que nuestra base de datos, en este caso MariaDB, un sistema relacional SQL, evolucione junto con el proyecto y se adapte a los cambios que surjan durante el desarrollo.

Finalmente, antes de empezar con el desarrollo de las historias de usuario, se realizará la *Spike 0*[12]. Este *Spike* es un elemento destinado a recopilar información y obtener el aprendizaje necesario para abordar incertidumbres y permitir la toma de decisiones en el desarrollo del producto sin una implementación inmediata. En este caso, se utilizará para investigar cómo conectar las tecnologías que se utilizarán para el desarrollo. De esta manera, se obtendrá un punto de partida sólido para comenzar con los sprints.

## 2.3. Planificación

### 2.3.1. Priorización

La priorización de las tareas ha sido un aspecto crucial en la planificación de cada uno de los sprints. Dado que las necesidades del proyecto no estaban claramente establecidas desde el principio y muchas de ellas han ido surgiendo durante la implementación, la discusión y decisión sobre qué tareas debían tener prioridad ha sido una parte integral de cada sesión de planificación del sprint.

En estas sesiones de planificación, se analizaban junto con el supervisor las tareas pendientes y se decidía cuales eran las fundamentales para alcanzar los objetivos y para poder continuar con el desarrollo del producto. A cada una de las tareas se les asignaba un nivel de prioridad y una estimación en puntos de historia para poder valorar cuales asignar a cada sprint. Para hacer esta estimación se definió un sistema de puntos en la sucesión de Fibonacci[18], de este modo las tareas iban puntuadas desde un 1 para aquellas más sencillas o de menor duración y 21 para las más complejas o que requerían más tiempo de desarrollo.

### 2.3.2. Spike 0

Antes de empezar con los sprints, durante las dos primeras semanas se llevaron a cabo dos tareas:

- Profundizar en las metodologías de programación eficiente leyendo el libro de Clean Code proporcionado por la empresa, con el objetivo de integrar estos conceptos y técnicas en las prácticas cotidianas de desarrollo, mejorando así la calidad del código y la mantenibilidad del software.
- Se realizó la Spike 0, que consiste en poner en marcha el proyecto con las tecnologías conectadas y una funcionalidad básica para verificar que todo funciona correctamente. Esta etapa es crucial para establecer una base sólida sobre la cual se desarrollarán las funcionalidades posteriores. Además permite familiarizarse con el entorno, tecnologías y librerías de trabajo. En cuanto a la base de datos, dado que era la segunda vez que se abordaba el proyecto, esta ya estaba creada, pero se eliminaron todas las entidades y relaciones existentes para empezar de nuevo.

### 2.3.3. Pila del producto inicial

El primer paso para el desarrollo del simulador de medidas para Smart Cities fue definir la pila del producto inicial, que se muestra en la figura 2.2.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | Todo | auth | very high |
| HU02 - Mostrar dispositivos del simulador | Todo | devices | very high |
| HU03 - Mostrar detalles de un dispositivo | Todo | devices | medium |
| HU04 - Mostar sensores de un dispositivo | Todo | sensors | very high |
| HU05 - Programar sensor para que genere medidas simuladas | Todo | simulation | very high |
| HU06 - Consultar medidas generadas por un sensor | Todo | sensors | very high |

Figura 2.2: Pila del producto inicial.

Esta pila contiene los requisitos que se consideraron imprescindibles para poder empezar a generar datos simulados de sensores de diferentes dispositivos. Estos requisitos se definieron junto con el supervisor de las prácticas, después de familiarizarme con el entorno de trabajo y la plataforma de IoTsens, que se utilizó como base para el proyecto.

### Planificación de la primera pila del sprint

Así pues, una vez definida la pila del producto inicial, se asignaron para este primer sprint las primeras cuatro historias, de HU01 a HU04, ya que se consideró que conformaban el primer paso para poder empezar a intentar simular medidas para los sensores.

### Ejemplo de definición de HU

Las historias de usuario son herramientas fundamentales en el desarrollo ágil de software, que sirven para capturar requisitos de software desde la perspectiva del usuario. Son breves, centradas en el valor y permiten a los equipos entender y construir funcionalidades que los usuarios realmente necesitan.

A continuación, se muestra, a modo de ejemplo, la definición de la HU02 asignada para este primer sprint. Para ello se detalla una descripción, dos posibles escenarios, el DoD (Definition of Done) que definiría el resultado esperado en la parte gráfica y la concreción de las tareas de alto nivel a realizar para implementar la historia. El resto de historias de usuario tienen un análisis similar.

**HU02 - Mostrar dispositivos del simulador**

**Descripción**

Como usuario quiero poder ver los dispositivos del simulador para poder interactuar con ellos.

**Escenarios**

E1 - válido - Hay dispositivos en el simulador

- Given: hay dispositivos importados
- When: se consultan los dispositivos del simulador
- Then: se muestran los dispositivos del simulador

E2 - válido - No hay dispositivos en el simulador

- Given: no hay dispositivos importados
- When: se consultan los dispositivos del simulador
- Then: se muestra un loader

**DoD**

Al visualizar los dispositivos importados en el simulador se muestran los dispositivos en una tabla o un loader si no hay ninguno importado.

**Tareas de implementación**

- Crear tabla correspondiente en base de datos
- Crear la entidad en el backend para relacionar la clase de java con la tabla de base de datos
- Crear el DAO (Data Access Object) para definir la sentencia SQL de obtención de los dispositivos
- Crear endpoint en el backend para obtener la información de base de datos
- Crear el servicio en el frontend para hacer la petición al backend para obtener los dispositivos almacenados
- Crear el componente de angular para mostrar los dispositivos en una tabla clicable

## 2.4. Seguimiento

En cuanto a la duración de las prácticas, es importante destacar que se iniciaron en el primer semestre, un periodo en el que la carga de trabajo docente era más alta de lo normal para la estancia en prácticas. Como resultado, la jornada se estableció en 4 horas. Esto hizo que el periodo de prácticas abarcara un total de 4 meses, incluyendo una interrupción de casi 3 semanas para los exámenes finales de enero, extendiendo la duración de la estancia en prácticas más allá de lo común.

Por otro lado, como ya se ha detallado, se ha adoptado la metodología Scrum para este Trabajo de Fin de Grado, con sprints de dos semanas de duración, lo que permitirá un desarrollo iterativo e incremental del simulador. Esta estructura ha resultado en un total de 7 sprints.

Durante estos sprints se generará una pila independiente en la que se detallarán las tareas necesarias a realizar en cada una de las historias de usuario. Una vez finalizado el sprint, todas aquellas historias de usuario que no se hayan podido acabar se devolverán al product backlog para ser completadas en futuras iteraciones.

Tal y como se ha mencionado, el producto a desarrollar tenía un alcance y una especificación limitada, lo que suponía un alto grado de incertidumbre y adaptabilidad. Evidencia de esto es la reducida cantidad de requisitos de la pila del producto inicial vista anteriormente en la sección 2.3.3. Debido a esta incertidumbre, no hay una planificación suficientemente clara a futuro. Sin embargo, es en cada sprint en el que se realizaban, a parte de la definición de la pila del sprint, las actividades de análisis de los requisitos, diseño de la solución, implementación de las funcionalidades, pruebas y validación y entrega del incremento del producto. No obstante, estas actividades no siempre se realizaban de forma secuencial o completa, sino que se adaptaban a las características y circunstancias de cada sprint, pero también a los resultados y conclusiones de los previos.

Por tanto, en cada sprint también se iba aumentando la pila del producto con requisitos completamente nuevos, que surgían a partir de la interacción con el supervisor de prácticas y otros desarrolladores que presentaban sus necesidades acerca del simulador. Estos nuevos requisitos se incorporaban a la pila del producto y se priorizaban según su importancia y urgencia, ya fuera para hacerlos en el propio sprint en el que se añadían o en posteriores.

Así pues, en el capítulo 3 se detallará el desarrollo de cada uno de los sprints, así como los resultados y las lecciones aprendidas en cada uno de ellos. También se mostrará la evolución de la pila del producto a lo largo de cada uno de los sprints, desde la inicial hasta la final.

## 2.5. Estimación de costes

En las siguientes secciones, se a va a realizar un análisis sobre los recursos empleados para desarrollar el proyecto, con una estimación de los costes de cada uno de ellos.

### 2.5.1. Recursos humanos

En cuanto a la partida de recursos humanos, para el desarrollo de este proyecto, se han visto involucrados los siguientes perfiles:

- Jefe del proyecto: el supervisor asignado a las prácticas actúa como el jefe del proyecto y se considera como perfil de desarrollador senior. Algunas tareas se han delegado a otros integrantes senior del departamento de software de la empresa, pero a efectos de coste se computarán como un único perfil.
- Desarrollador: el alumno actúa como único integrante del equipo de desarrollo para el simulador, considerándose como perfil de programador junior, por lo que se le aplica la tarifa de este perfil y no la remuneración recibida durante las prácticas.
- Personal de sistemas: para poder llevar a cabo el desarrollo del proyecto se ha requerido la preparación de equipos e instalación de programas, con lo que debe tenerse en cuenta las horas dedicadas por el personal de sistemas.
- Scrum Manager: al realizar reuniones diarias para el seguimiento de los distintos proyectos que se están llevando a acabo dentro del departamento, se tendrá el tiempo dedicado del responsable de estas reuniones como perfil Engeenering Manager.

Una vez identificados los perfiles del personal involucrado de una forma u otra en el desarrollo del proyecto, se debe hacer una estimación del coste/hora de cada uno de ellos y del tiempo dedicado. Por un lado, se estima media hora diaria dedicada bien por el supervisor de las prácticas o bien por algún otro miembro del equipo de desarrolladores del departamento.

Además, se debe incluir la parte proporcional de la media hora diaria que dedicaba el Scrum Manager en las reuniones diarias. Se dividirá está dedicación entre los 14 participantes habituales de dichas reuniones.

Por último, se hace una estimación del tiempo dedicado por el personal de sistemas en la instalación y configuración del equipo y el software requerido.

El coste de estimado en recursos humanos se muestra en el cuadro 2.1.

| **Concepto** | **Coste/hora(€)** | **Horas** | **Coste total(€)** |
|---|---|---|---|
| Jefe del proyecto | 30 | 37,5 | 1.125 |
| Desarrollador | 20 | 300,0 | 6.000 |
| Personal de sistemas | 25 | 5,0 | 125 |
| Scrum Manager | 30 | 2,7 | 81 |
| **Total estimado** | | | **7.331** |

Cuadro 2.1: Estimación de costes en recursos humanos del proyecto.

### 2.5.2. Recursos tecnológicos

El software empleado para el desarrollo del producto ha sido detallado en apartados anteriores. Toda esa *suite* de aplicaciones es *Open Source*, por lo que no ha tenido ningún coste relacionado. La excepción ha sido la suscripción a IntelliJ IDEA Ultimate, que se ha mantenido durante los cuatro meses de prácticas.

En cuanto al hardware, la empresa ha proporcionado un ordenador portátil, dos monitores, un teclado y un ratón. Este equipamiento tiene una vida útil estimada de dos años, así que el coste debe repartirse entre los años de reposición e imputarse al proyecto solo el correspondiente al tiempo en que se han realizado las prácticas, en este caso 4 meses.

El coste de estimado en recursos tecnológicos se muestra en el cuadro 2.2.

| **Concepto** | **Coste/mes(€)** | **Meses** | **Coste total(€)** |
|---|---|---|---|
| Software (IntellJ IDEA Ultimate) | 59,90 | 4 | 239,60 |
| Ordenador | 18,71 | 4 | 74,84 |
| Monitores(2) | 10,42 | 4 | 41,68 |
| Accesorios | 5,42 | 4 | 21,68 |
| **Total estimado** | | | **377,80** |

Cuadro 2.2: Estimación de costes tecnológicos del proyecto.

### 2.5.3. Otros

Además de los costes directos ya detallados, la empresa ha invertido en recursos adicionales para la capacitación. Inicialmente, se facilitó el acceso al libro Clean Code para fomentar las mejores prácticas de programación. Adicionalmente, previo al inicio de las prácticas, se adquirió un curso de Udemy centrado en el framework Angular, proporcionando una base sólida para comenzar el desarrollo del producto.

El coste de estimado en otros recursos se muestra en el cuadro 2.3.

| **Concepto** | **Coste(€)** |
|---|---|
| Libro Clean Code | 19,65 |
| Curso Angular | 219,99 |
| **Total estimado** | **239,64** |

Cuadro 2.3: Estimación de otros costes del proyecto.

### 2.5.4. Costes totales

La estimación de los costes totales del proyecto incluye la suma de los recursos humanos, tecnológicos y de formación. En resumen, se consideran las horas de trabajo del personal involucrado, el coste de las herramientas de software y hardware proporcionadas, y la inversión en material educativo.

En cuanto a la estimación de los costes indirectos asociados al desarrollo del producto, se han calculado como el 20 % del coste total de los recursos humanos.

La cifra final, tal y como se muestra en el cuadro 2.4, proporcionará una visión global del presupuesto del desarrollo del simulador.

| **Concepto** | **Coste(€)** |
|---|---|
| Recursos Humanos | 7.331,00 |
| Recursos Tecnológicos | 377,80 |
| Otros | 239,64 |
| Costes indirectos | 1.466,20 |
| **Total estimado** | **9.414,64** |

Cuadro 2.4: Estimación de costes del proyecto.

## 2.6. Riesgos

La gestión de riesgos es un aspecto crucial en cualquier proyecto de desarrollo de software. En este proyecto, se han identificado varios riesgos potenciales que podrían afectar a su éxito. Estos riesgos se agrupan en cuatro categorías principales:

- El entorno del proyecto presenta ciertos desafíos, fundamentalmente que los objetivos del proyecto están poco especificados, con solo un objetivo principal definido. Esto se debe a que se ha intentado abordar el proyecto anteriormente sin éxito, lo que ha llevado a la empresa a definir solo el objetivo principal e ir avanzando poco a poco conforme se vayan alcanzando metas.

- El equipo de desarrollo es pequeño y no presenta problemas. Sin embargo, la falta de experiencia es un factor de riesgo. Para mitigarlo, se decide realizar un curso de Angular antes de comenzar las prácticas, intentado así asegurar una base más sólida para el desarrollo.
- El producto tiene un alcance indefinido y está sujeto a una evolución continua, lo que no permite fijar una meta a priori. Para manejar esto, se aplicará la metodología ágil Scrum, que permite reajustes continuos y la incorporación de nuevos requisitos basados en el feedback recibido.
- Los usuarios del producto son miembros del equipo de desarrollo de software de la empresa puesto que el producto es una aplicación para uso interno. Las expectativas son altas, afortunadamente, los usuarios demuestran comprensión, lo que puede ayudar a mitigar los riesgos asociados. Además, su participación activa en el desarrollo reduce el riesgo de no cumplir con estas expectativas y facilitan la alineación del producto con las necesidades reales.

En las siguientes secciones, se detallarán los riesgos y se proporcionarán planes de prevención y corrección para cada uno de ellos.

### 2.6.1. Análisis de riesgos

En la tabla 2.5 se definen los riesgos potenciales identificados que pueden influir negativamente en el éxito del proyecto.

| **ID** | **Descripción** |
|---|---|
| R01 | Objetivos poco definidos |
| R02 | Estimación inexacta de las historias de usuario |
| R03 | Sobrecarga de historias de usuario al sprint |
| R04 | Implementación inconsistente |
| R05 | Necesidad de adaptación continúa de la base de datos |
| R06 | Intento de implementación previo fallido |

Cuadro 2.5: Definición de riesgos.

Una vez definidos los riesgos, en la tabla 2.6, se detalla el análisis de los mismos.

| | |
|---|---|
| R01 | **Magnitud**: Alta<br>**Descripción**: Solo se define el objetivo principal, se irán definiendo nuevos objetivos intermedios a medida que avanza el proyecto, lo que puede llevar a cambios significativos en su alcance.<br>**Impacto**: Incertidumbre en el progreso del proyecto y aumento en la carga de trabajo.<br>**Indicadores**: Se añaden y modifican muchas historias de usuario partiendo de una pequeña pila del producto inicial. |
| R02 | **Magnitud**: Media<br>**Descripción**: Las historias de usuario se extienden más de lo previsto, afectando la planificación del sprint.<br>**Impacto**: Retrasos en las entregas y posible sobrecarga de trabajo en sprints futuros.<br>**Indicadores**: Incremento en el tiempo de finalización de las historias de usuario y ajustes frecuentes en la planificación del sprint. |
| R03 | **Magnitud**: Baja<br>**Descripción**: Las historias de usuario no se asignan de manera efectiva durante la planificación del sprint, lo que puede afectar la distribución del trabajo.<br>**Impacto**: Carga de trabajo del sprint mal equilibrada.<br>**Indicadores**: Tareas pendientes acumuladas y desigualdad en la distribución de las tareas. |
| R04 | **Magnitud**: Media<br>**Descripción**: El patrón que sigue la implementación de las funcionalidades evoluciona conforme se va adquiriendo experiencia.<br>**Impacto**: Necesidad de refactorización y posibles retrasos en la finalización de las tareas del sprint.<br>**Indicadores**: Código fuente con inconsistencias. |
| R05 | **Magnitud**: Media<br>**Descripción**: Necesidad de un sistema de control de versiones robusto para la base de datos, para evitar conflictos y pérdida de datos.<br>**Impacto**: Riesgo de perder cambios importantes y dificultad para mantener la integridad de la base de datos.<br>**Indicadores**: Conflictos en la base de datos y problemas en la migración de datos. |
| R06 | **Magnitud**: Baja<br>**Descripción**: El intento previo de implementación no tuvo existo, con lo que pueden existir cierta reticencia a abordar de nuevo el proyecto.<br>**Impacto**: Desmotivación del equipo y posible repetición de errores pasados.<br>**Indicadores**: Falta de aprendizaje de experiencias anteriores y repetición de patrones de fallos. |

Cuadro 2.6: Análisis detallado de riesgos.

Finalmente, la tabla 2.7 propone las medidas a implementar para prevenir y mitigar los impactos de los riesgos identificados.

| **ID** | **Plan de Prevención/Acción** | **Plan de Corrección/Contingencia** |
|---|---|---|
| R01 | Emplear una metodología ágil capaz de adaptarse a los cambios, realizando reuniones para valorar el futuro del proyecto en base al estado actual. | Hacer una reunión extraordinaria para volver a encauzar los objetivos. |
| R02 | Crear historias de usuario de poco alcance, reunión de planificación del sprint y estimación de puntos de historia. | Dividir la historia de usuario en varias o redefinir su alcance. |
| R03 | Valorar la velocidad de implementación de puntos de historia del equipo. | Reducir o ampliar pila del sprint durante su desarrollo, añadiendo nuevas historias de usuario a la pila del producto para futuros sprints o no incluirlas en el alcance |
| R04 | Mayor grado de formación antes de empezar el desarrollo. | Refactorizar el código. |
| R05 | Usar herramienta de gestión de versiones de base de datos. | Revertir versionas hasta un punto válido y redefinir diseño físico. |
| R06 | Fijar un objetivo básico y apostar por la utilidad del producto. | Redefinir objetivos en función del éxito obtenido. |

Cuadro 2.7: Planes de prevención y corrección de riesgos.

# Capítulo 3

# Desarrollo del proyecto por sprints

Como ya se ha comentado en repetidas ocasiones, excepto el objetivo principal, a priori no se habían definido unos objetivos claros a desarrollar. Debido a la evolución constante del producto, en cada sprint se han evaluado los requisitos planteados hasta ese momento, se ha valorado su idoneidad y se han realizado las actividades requeridas para satisfacer las necesidades de la empresa. Por tanto, en cada sprint se ha modificado la pila del producto adecuándola a las nuevas necesidades, añadiendo nuevas historias que se han priorizado y asignado a cada sprint según el valor aportado, la urgencia de abordar los requisitos y sus dependencias. A lo largo de este capítulo se detallan cada uno de los sprints realizados y las actividades llevadas a cabo.

Al final de cada sprint, se ha realizado una demostración al supervisor de las prácticas y, en algunos casos, a otros miembros del equipo de desarrollo de la empresa. En estas demostraciones, se ha recogido el feedback y se han planteado los objetivos del siguiente sprint. Esta tarea se ha repetido en todos los sprints, por lo que no se mencionará en cada uno de ellos a menos que sea relevante.

En el capítulo 4 se presentan los resultados finales de la implementación de todos los sprints, para evitar redundancias y facilitar la comprensión.

## 3.1. Sprint 1

Las secciones siguientes detallan la evolución de la pila del producto y el proceso de implementación de las historias de usuario durante el sprint, incluyendo los éxitos y los errores cometidos.

### 3.1.1. Pila del producto

Al analizar más profundamente la pila inicial, mostrada en la figura 3.1, se decidió descomponer la HU01, relativa a la autenticación en general, en tres historias de gestión de usuarios más específicas: HU07, HU08 y HU09, referentes a iniciar sesión, detalles de la cuenta y cerrar sesión, respectivamente.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | Todo | auth | very high |
| HU02 - Mostrar dispositivos del simulador | Todo | devices | very high |
| HU03 - Mostrar detalles de un dispositivo | Todo | devices | medium |
| HU04 - Mostar sensores de un dispositivo | Todo | sensors | very high |
| HU05 - Programar sensor para que genere medidas simuladas | Todo | simulation | very high |
| HU06 - Consultar medidas generadas por un sensor | Todo | sensors | very high |

Figura 3.1: Pila del producto al inicio del proyecto.

La HU07 se consideró prioritaria, ya que era necesaria para establecer la comunicación entre el simulador y la base de datos central de IoTsens. Las otras dos historias, HU08 y HU09, aunque no tan importantes, se propusieron también para este sprint. De este modo, se dejaría ya concluida la parte de autenticación del usuario.

La pila del producto adecuada a estas modificaciones se puede ver en la figura 3.2, mientras que la pila del sprint con sus historias asignadas se muestra en la figura 3.3.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | Todo | auth | very high |
| HU02 - Mostrar dispositivos del simulador | Todo | devices | very high |
| HU03 - Mostrar detalles de un dispositivo | Todo | devices | medium |
| HU04 - Mostar sensores de un dispositivo | Todo | sensors | very high |
| HU07 - Iniciar sesión | Todo | auth | very high |
| HU08 - Detalles de la cuenta | Todo | auth | Very low |
| HU09 - Cerrar sesión | Todo | auth | Very low |
| HU05 - Programar sensor para que genere medidas simuladas | Todo | simulation | very high |
| HU06 - Consultar medidas generadas por un sensor | Todo | sensors | very high |

Figura 3.2: Pila del producto al inicio del primer sprint.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | Todo | auth | very high |
| HU02 - Mostrar dispositivos del simulador | Todo | devices | very high |
| HU03 - Mostrar detalles de un dispositivo | Todo | devices | medium |
| HU04 - Mostar sensores de un dispositivo | Todo | sensors | very high |
| HU07 - Iniciar sesión | Todo | auth | very high |
| HU08 - Detalles de la cuenta | Todo | auth | Very low |
| HU09 - Cerrar sesión | Todo | auth | Very low |

Figura 3.3: Pila del sprint 1.

### 3.1.2. Desarrollo del sprint

Una vez finalizada la formación en las distintas herramientas y conseguido un esqueleto de la aplicación durante la Spike 0, se comenzó a trabajar en el primer sprint.

El primer objetivo de este sprint fue la gestión de dispositivos. Para ello, se creó una tabla en la base de datos para almacenar los dispositivos y sus propiedades. Tras crear la estructura, se introdujeron manualmente algunos dispositivos en el simulador, ya que se pretendía listarlos y, por tanto, esos datos de prueba servirían para verificar la funcionalidad y las conexiones. Además, permitiría centrarse en afianzar el uso de la interfaz de usuario.

El esquema del diseño conceptual de la base de datos en este primer sprint se muestra en la figura 3.4.

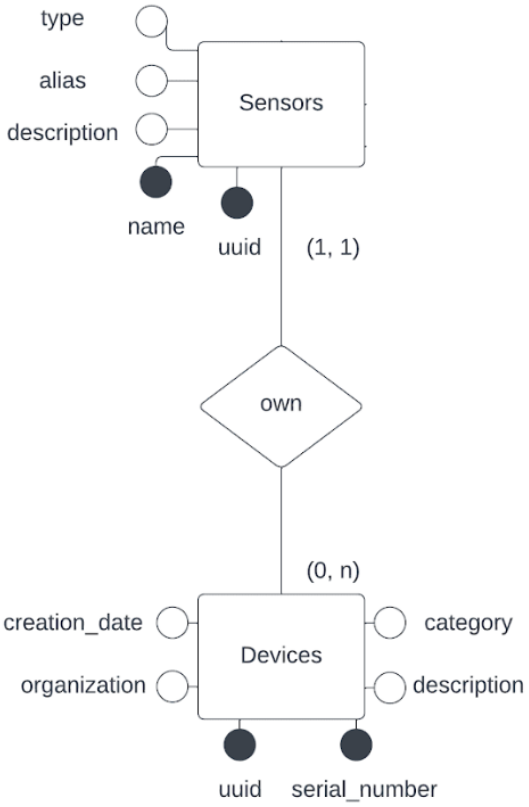



Figura 3.4: Diseño conceptual de la base de datos en el sprint 1.

Para alcanzar este objetivo, se diseñaron dos páginas en Angular y se definieron las rutas que permitían acceder a ellas: */devices* para listar los dispositivos y */devices/details* para ver sus detalles y sensores. Dada la naturaleza de los datos que se querían mostrar, se decidió que la lista estuviera presentada en formato de tabla y que fuera interactiva, usando PrimeNG para crear este componente. De este modo, al seleccionar un dispositivo, se muestran sus detalles y todos los sensores vinculados. A su vez, la visualización de los sensores también se realizó en formato de tabla, y se creó una entidad adicional en la base de datos para almacenar la información vinculada a los sensores, cuyos datos, de nuevo, se insertaron manualmente.

A modo de ejemplo, en la figura 3.5 se puede ver la página de detalles de un dispositivo junto a sus sensores.

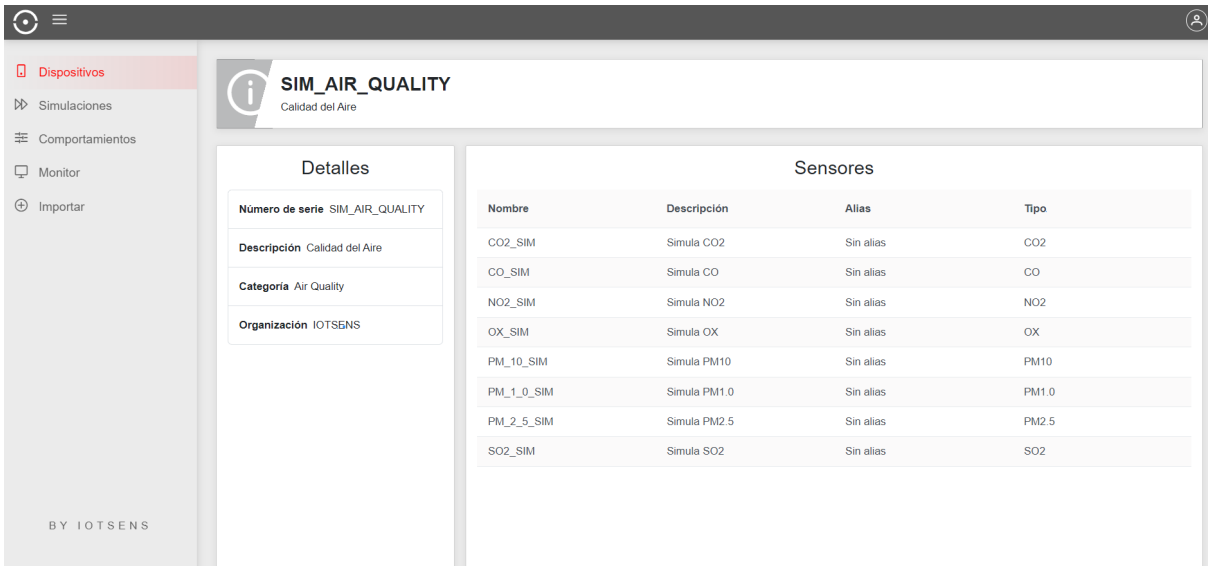



Figura 3.5: Página de detalles de un dispositivo.

Las interacciones con la base de datos mencionadas anteriormente no se realizaron directamente en MariaDB, sino que se utilizaron scripts de versiones en Flyway. Cada script de versión incluía o bien la creación de una tabla, o bien inserciones de datos. En el código 3.1 se muestra, a modo de ejemplo, uno de los scripts creados.

Código 3.1: Creación de la tabla sensors.

```
1 CREATE TABLE sensors (
2     uuid VARCHAR(255) PRIMARY KEY,
3     name VARCHAR(255) NOT NULL UNIQUE,
4     description VARCHAR(255) NOT NULL,
5     alias VARCHAR(255) NOT NULL,
6     type VARCHAR(50) NOT NULL,
7     device_uuid VARCHAR(255) NOT NULL,
8     FOREIGN KEY (device_uuid) REFERENCES devices(uuid) ON
        ↪ UPDATE CASCADE
9 );
```

Además, a pesar de que en ese momento no tenía apenas conocimiento sobre como funciona un sensor, el supervisor indicó que estos emiten una medida cada cierto intervalo de tiempo. Por tanto, se consideró oportuno experimentar con la creación de un programador (scheduler) de Spring[14], que se utiliza para ejecutar una tarea cada cierto intervalo de tiempo. Este programador de prueba generaba un dato completamente aleatorio cada minuto, intentando reproducir el comportamiento de un sensor real y simulando así las primeras mediciones.

Después de mostrar los dispositivos y sensores e implementar el primer programador que generaba medidas aleatorias, se implementó una tabla para visualizar las medidas de un sensor. De forma análoga al funcionamiento de la tabla de dispositivos, al seleccionar un sensor se mostraban sus medidas. Esta tabla, al igual que las anteriores, se desarrolló con la ayuda de PrimeNG.

No obstante, una vez implementada dicha tabla, se observó que no era cómodo y que una alternativa más atractiva sería mostrar dichos valores en una gráfica. Para ello, se creó una nueva sección dentro del menú del simulador, llamada *Monitor*, en la que se implementó una gráfica que mostraba todas las medidas actuales. Para generar este componente visual se empleó la biblioteca HighCharts[8].

El siguiente objetivo a abordar, fue la gestión de usuarios. Se buscaba permitir a los usuarios registrados en el sistema de la empresa iniciar y cerrar sesión para acceder a la aplicación, así como visualizar los detalles de su cuenta, alineándose con el comportamiento del resto de aplicaciones de la suite de IoTsens. Para implementar esta funcionalidad, se diseñó una página con un formulario de inicio de sesión que aparece siempre al entrar al simulador por primera vez, independientemente de la ruta a la que se accediese (dispositivos o sensores). Una vez que el usuario iniciaba sesión, ya podía utilizar el simulador. Pude probar esta funcionalidad usando mis credenciales de empresa para acceder al simulador.

### 3.1.3. Detalles de la implementación

Como se ha detallado anteriormente, las historias de usuario asignadas para este sprint fueron: HU02, HU03, HU04, HU07, HU08 y HU09, enfocadas en la gestión de dispositivos, sensores y usuarios.

A continuación, se resume de cada una de ellas y se valoran los resultados obtenidos:

- **HU07, HU08, HU09 - Gestión de usuarios:** para estas historias, se empleó como referencia una implementación existente de la aplicación central de IoTsens, con el fin de mantener la coherencia y la simplicidad. Sin embargo, hubo algunos problemas para llevar a cabo la autenticación y la conexión a la base de datos, lo que provocó cierto retraso, pero finalmente se logró completar la HU07 con éxito. Las historias HU08 y HU09, que se encargaban de mostrar detalles de la cuenta y cerrar sesión, respectivamente, se pospusieron ya que su implementación resultó ser más difícil de lo esperado inicialmente y no eran cruciales para seguir desarrollando el simulador.

- **HU02, HU03 - Gestión de dispositivos:** como se ha comentado anteriormente, para estas historias se implementaron unas tablas para mostrar los dispositivos y sus detalles, dándose por alcanzados los objetivos planteados.

- **HU04 - Gestión de sensores:** se simplificó el proceso datos de prueba introducidos manualmente. Además, como se tenían que usar tablas de nuevo, valorando pros y contras, decidí generalizar el componente creado para las historias anteriores con el fin de poder reutilizarlo.

Después de los contratiempos iniciales con la parte de autenticación, el resto de las historias se implementaron rápidamente. Por ello, se añadieron al sprint las HU05 y HU06, que inicialmente no estaban previstas, referidas a la programación y consulta de medidas simuladas.

- **HU05 - Programar sensor:** esta historia se hizo solo de forma parcial, para explorar las posibilidades y aprender más sobre la simulación, probando a crear un scheduler de Spring. Con esta funcionalidad lograda, se consideró suficiente para este sprint y la historia de usuario se devolvió a la pila del producto para ser completada más adelante.
- **HU06 - Consultar medidas:** inicialmente se mostraban todas las medidas del único sensor, definido manualmente, en una tabla. Sin embargo, al observarse que esta forma de consultar las medidas no era práctica, se empezó a trabajar en la visualización gráfica de los datos. Al final del sprint, la HU06 volvió al product backlog para ser acabada en los siguientes sprints y se añadió una nueva historia de usuario relativa a visualizar los valores en una gráfica.

La pila del producto junto con el tablero Kanban tras este primer sprint se puede ver en las figuras 3.6 y 3.7.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | deferred | auth | very high |
| HU05 - Programar sensor para que genere medidas simuladas | In Progress | simulation | very high |
| HU06 - Consultar medidas generadas por un sensor | In Progress | sensors | very high |
| HU08 - Detalles de la cuenta | deferred | auth | Very low |
| HU09 - Cerrar sesión | deferred | auth | Very low |
| HU10 - Mostrar en una gráfica medidas generadas por un sensor | deferred | generated data | medium |
| HU11 - Definir un plan de simulación periódico | Todo | simulation | very high |
| HU12 - Definir un plan de simulación de estrés | Todo | simulation | very high |
| HU13 - Asignar sensores a un plan de simulación | Todo | simulation | very high |
| HU14 - Consultar planes de simulación definidos | Todo | simulation | high |
| HU15 - Filtrar planes de simulación definidos por tipo | Todo | simulation | Very low |
| HU16 - Modificar planes de simulación | Todo | simulation | high |
| HU17 - Eliminar planes de simulación | Todo | simulation | medium |

Figura 3.6: Pila del producto al final del sprint 1.

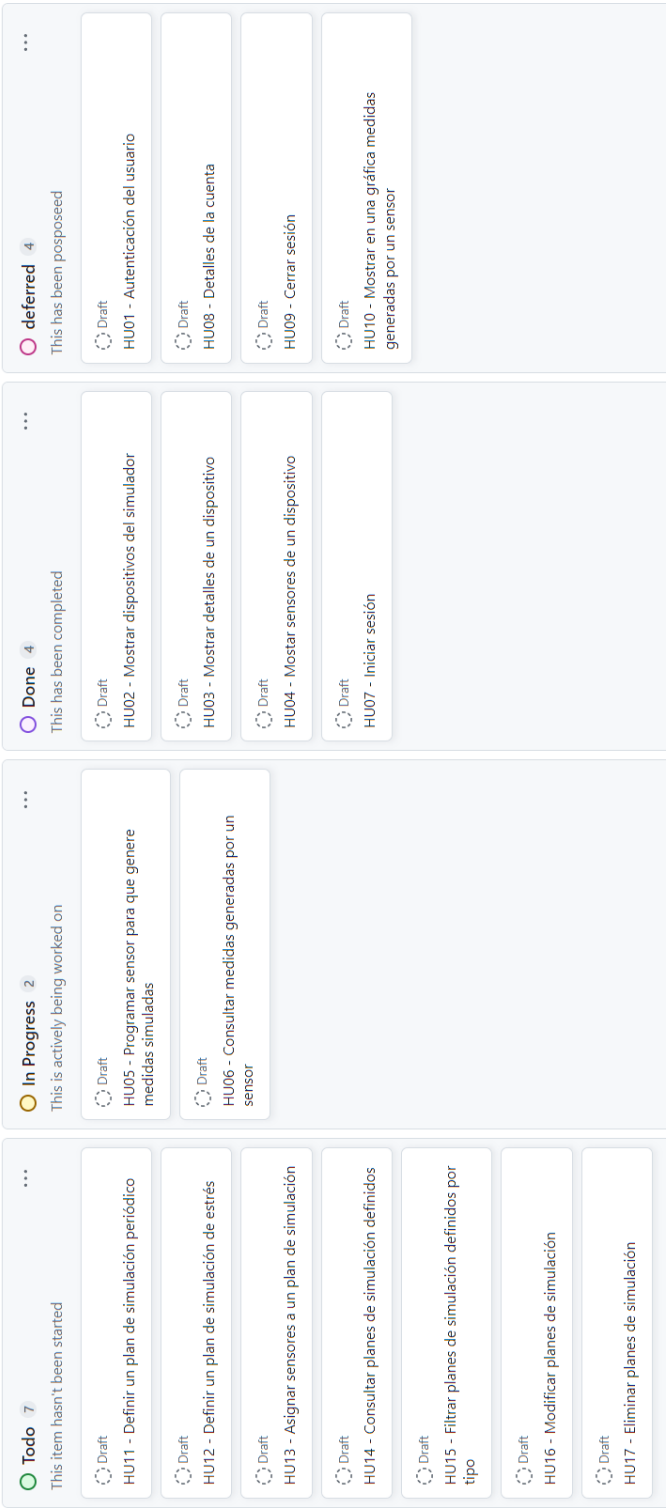



Figura 3.7: Estado del Kanban al final del sprint 1.

## 3.2. Sprint 2

Al principio de este sprint y tras conseguir las primeras medidas generadas cada minuto totalmente aleatorias en el sprint anterior, era vital para continuar con el desarrollo plantearnos como simular un sensor y la relevancia de los schedulers para conseguirlo.

### 3.2.1. ¿Cómo simular un sensor?

Es crucial destacar que los sensores realizan mediciones del entorno en intervalos de tiempo definidos, conocidos como frecuencia de muestreo. Por ejemplo, si un sensor tiene una frecuencia de muestreo de un minuto, esto significa que capturará una muestra del entorno y la transmitirá cada minuto. Esta capacidad de especificar la frecuencia de muestreo permite un control preciso sobre la cantidad y la regularidad de los datos recopilados.

Por lo tanto, para simular el comportamiento de un sensor real, hay que tener en cuenta la frecuencia con la que se desea tomar una muestra del entorno.

### 3.2.2. ¿Cómo el simulador da respuesta a los requisitos?

Una vez que está claro que se debe poder definir la frecuencia de muestreo de un sensor, hay que definir como se va a permitir esta acción en el simulador.

Las dos opciones posibles son:

- Definir esta frecuencia como un atributo de cada uno de los sensores.
- Definir esta frecuencia en una abstracción, para que todos los sensores pertenecientes a ella tomen esa frecuencia.

El supervisor de las prácticas indicó que un sensor real funciona de acuerdo con la primera opción. Sin embargo, se me otorgó autonomía para explorar otras formas de implementación.

Evaluando posibles alternativas con el objetivo de generar datos simulados a una frecuencia determinada, desarrollé como solución la idea de diseñar planes que avisaran a los sensores asignados cuando les tocara simular un dato. Esta idea se inspiró en el patrón del observador[2], que consiste en que un sujeto notifica a sus observadores de algún cambio de estado. De esta manera, los sensores no tendrían que llevar su propio temporizador. Además, esta solución se adaptaba bien a los schedulers de Spring, ya que permitía simular datos para varios sensores con un solo scheduler por cada frecuencia de muestreo. Si no fuera así, habría que crear un scheduler para cada sensor. El supervisor aprobó esta idea y se procedió a implementarla.

En cuanto al nombre de estas abstracciones, se decidió llamarlas *planes de simulación periódicos*.

A continuación, además de simular las medidas de forma periódica, me pareció interesante poder simular un gran volumen de datos para hacer pruebas de carga. Por lo que pensé en diseñar otro tipo de plan de simulación que en vez de agrupar los sensores por frecuencia, se agruparan en un plan que les indicara el volumen de datos que debían generar.

De esta forma, habría dos tipos de planes de simulación: uno periódico, que establece la frecuencia de muestreo con la que se quiere simular una medida de sus sensores asignados, y otro que especifica la cantidad de datos a simular en un instante, útil para pruebas de estrés de las soluciones Smart City. A este último tipo de planes se le llamó *planes de simulación de estrés*.

En resumen, el simulador tendrá estos dos tipos de planes de simulación:

- Planes de simulación periódicos: establecerán la regularidad con la que los sensores asignados simularán una medida.
- Planes de simulación de estrés: definirán el volumen de medidas a simular por cada uno de los sensores asignados en un tiempo 0.

### 3.2.3. Pila del producto

Tras definir como simular un sensor es necesario actualizar la pila del producto para que refleje estas nuevas necesidades. Centrándonos en la pila de este sprint, la figura 3.8 muestra las historias de usuario asignadas.

| | Title | Status | Priority |
|---|---|---|---|
| 1 | HU05 - Programar sensor para que genere medidas simuladas | In Progress | very high |
| 2 | HU06 - Consultar medidas generadas por un sensor | In Progress | very high |
| 3 | HU11 - Definir un plan de simulación periódico | Todo | very high |
| 4 | HU12 - Definir un plan de simulación de estrés | Todo | very high |
| 5 | HU13 - Asignar sensores a un plan de simulación | Todo | very high |
| 6 | HU14 - Consultar planes de simulación definidos | Todo | high |
| 7 | HU15 - Filtrar planes de simulación definidos por tipo | Todo | Very low |

Figura 3.8: Pila del sprint 2.

La historia HU05, que era muy amplia, se consideró como una épica y se dividió en tres subhistorias tras el análisis y diseño realizado anteriormente. Estas nuevas historias consistirían en: definir un plan de simulación periódico, definir un plan de simulación de estrés y asignar sensores a un plan de simulación, HU011, HU12, HU13 respectivamente.

Además, se permitía consultar los planes de simulación y filtrarlos por su tipo (periódicos o de estrés).

Por último, como historias de usuario añadidas a la pila del producto, pero no contempladas para este sprint, se incluyó, tras comprobar su gran utilidad, una historia de usuario para mostrar las medidas simuladas en una gráfica. También se agregaron historias de usuario para gestionar los planes de simulación creados, es decir, editarlos y borrarlos.

### 3.2.4. Desarrollo del sprint

Al comienzo de este sprint, a pesar de no ser un requisito, y tras ver que se empleaban tablas en muchas páginas del simulador, consideré oportuno ampliar y mejorar el componente reutilizable diseñado en el sprint 1 para que fuera de uso más general y reusable. Este componente debía ser sumamente configurable y usar *lazy-load*[9] para manejar eficientemente los datos, debido al gran volumen de medidas generadas por las simulaciones. Para ello, esta nueva tabla solo cargaba las primeras filas y presentaba un desplazamiento vertical que permitía navegar por ella, cargando los datos justo antes de que se vayan a visualizar.

Posteriormente, para implementar los planes de simulación, fue necesario modificar la base de datos utilizando Flyway. Se creó una nueva tabla de planes de simulación con los campos comunes entre los dos tipos y, para cada uno de ellos, se añadió otra adicional. Además, los planes de simulación se tuvieron que relacionar con los sensores, tanto para definir las asignaciones como las medidas simuladas. Estas modificaciones sobre la base de datos se pueden observar en la figura 3.9.

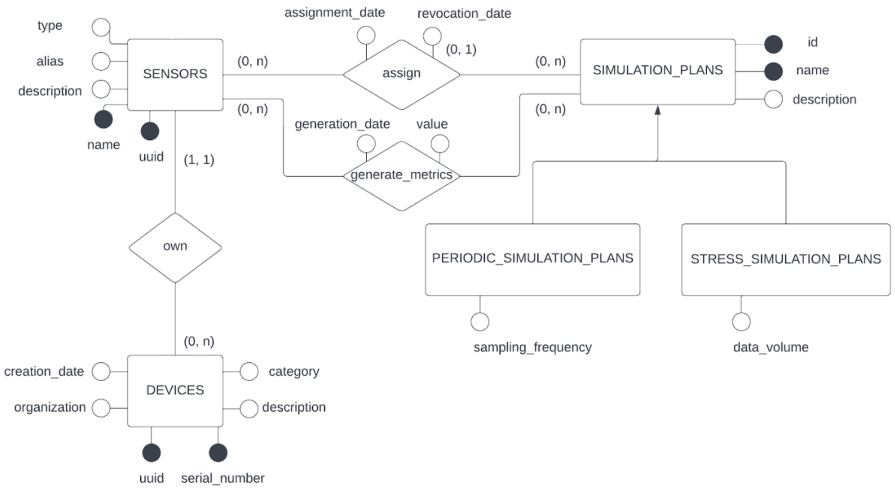



Figura 3.9: Diseño conceptual de la base de datos en el sprint 2.

En términos de implementación funcional, se añadió un nuevo elemento en el menú lateral del simulador, con las respectivas rutas, para esta sección de planes de simulación. Al acceder a este nuevo apartado, se visualizaba la lista de planes de simulación existentes actualmente en el simulador.

Por otro lado, para permitir filtrar, se habilitó una nueva funcionalidad para que al pulsar sobre el elemento del menú de simulaciones por segunda vez, estando previamente seleccionado, se mostrara otro menú en el lateral opuesto de la pantalla. Este menú permitiría filtrar la lista de planes por tipo.

A continuación, se añadió un botón para crear un plan de simulación. Este es un proceso complejo en el que se tiene que elegir el tipo, definir los atributos generales y particulares (frecuencia de muestreo en caso de planes de simulación periódicos y volumen de datos en el caso de ser del tipo de estrés) y asignar sensores. Por lo tanto, se optó por implementar un proceso paso a paso, proporcionando un resumen de las acciones a completar junto con la actual, para mejorar la usabilidad.

Para llevar a cabo este proceso, se crearon varias rutas, una para cada paso, y un servicio encargado de mantener todo el estado global. Finalmente, se añadieron unos botones para navegar entre los pasos y una funcionalidad para redirigir automáticamente a la página de inicio del proceso de crear un plan de simulación (selección del tipo del plan de simulación) si se intentaba acceder manualmente a alguna ruta del CRUD o si se recargaba la página del paso actual, con la intención de que no se quedase en un estado inconsistente.

En la figura 3.10 se puede ver el primero de estos pasos de creación de un plan de simulación donde se elije el tipo, junto con el *timeline* de todos los pasos de este proceso.

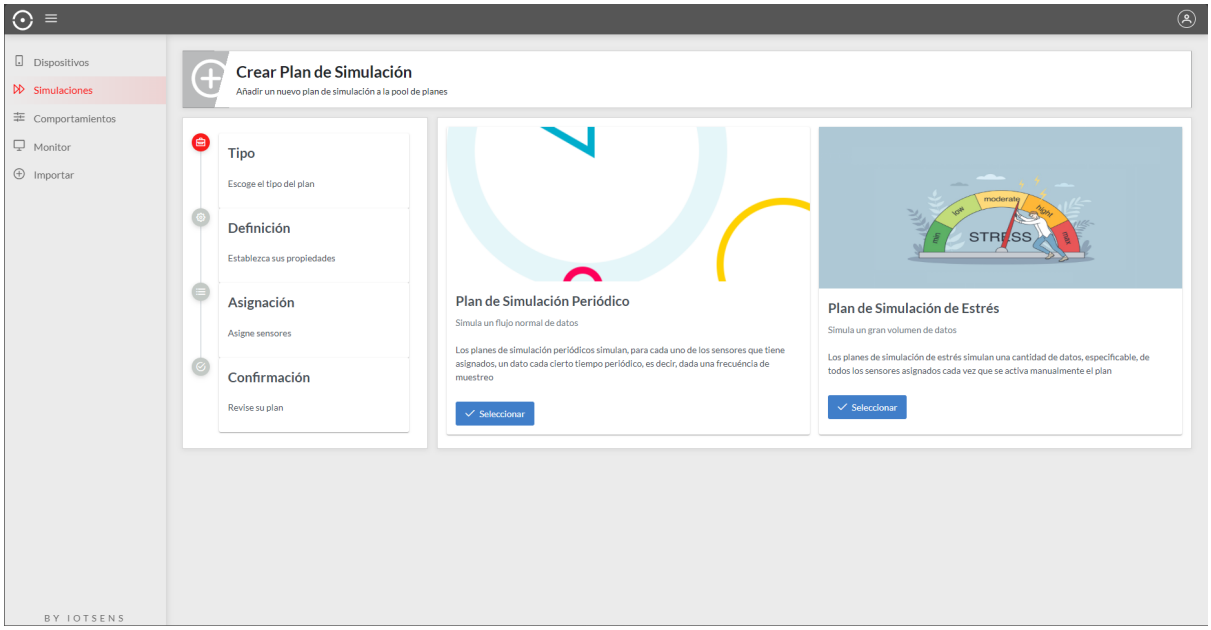



Figura 3.10: Primer paso de la creación de un plan de simulación.

Finalmente, se crearon algunos dispositivos y sensores de prueba en la base de datos. Esto eliminó la necesidad de introducirlos manualmente, como se hizo en el sprint anterior. De esta manera, se lograron unas comunicaciones y flujo de datos más realistas.

### 3.2.5. Detalles de la implementación

En este sprint, como se mencionó anteriormente, además de trabajar en las historias de usuario asignadas, se decidió crear un componente reutilizable para mostrar tablas en diferentes páginas. Esta tarea resultó ser bastante desafiante y consumió casi la mitad del tiempo del sprint.

En cuanto a las historias de usuario, se completó la HU06, que quedó pendiente del sprint anterior. Para ello, fue necesario cambiar el endpoint[13] en el backend para que consultara las medidas de un sensor específico en lugar de todas las del simulador.

Una vez concluida esta historia y el nuevo componente, quedaba abordar las siguientes:

- **HU11, HU12 y HU13 - Gestión de Planes de Simulación:** las historias HU11 y HU12 se completaron con éxito. Sin embargo, la HU13, que consistía asignar los sensores al plan de simulación, tras ver lo complicado que fue implementar los primeros pasos de este proceso, se pospuso hasta el siguiente sprint.
- **HU14, HU15 - Gestión de Planes de Simulación:** estas historias requerían implementar una tabla para ver todos los planes de simulación creados por el usuario y permitir filtrar los planes por tipo: periódicos, de estrés o todos (HU15). Lamentablemente, el filtro tenía un problema de usabilidad, ya que no funcionaba consistentemente al hacer clic sobre la sección de simulaciones del menú lateral. Ante las dificultades para encontrar una solución, se aplazó esta historia para futuros sprints y se devolvió a la pila del producto.

Después de la demostración, el supervisor de las prácticas sugirió que sería útil, al ver los detalles de un sensor, poder consultar el plan de simulación asignado. Así que, se añadió una nueva historia de usuario.

La pila del producto final y el Kanban de este sprint se pueden ver en las figuras 3.11 y 3.12 respectivamente.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | deferred | auth | very high |
| HU08 - Detalles de la cuenta | deferred | auth | Very low |
| HU09 - Cerrar sesión | deferred | auth | Very low |
| HU10 - Mostrar en una gráfica medidas generadas por un sensor | deferred | generated data | medium |
| HU05 - Programar sensor para que genere medidas simuladas | In Progress | simulation | very high |
| HU15 - Filtrar planes de simulación definidos por tipo | deferred | simulation | Very low |
| HU13 - Asignar sensores a un plan de simulación | Todo | simulation | very high |
| HU16 - Modificar planes de simulación | Todo | simulation | high |
| HU17 - Eliminar planes de simulación | Todo | simulation | medium |
| HU18 - Mostrar plan asignado a sensor | Todo | sensors | low |

Figura 3.11: Estado de la pila del producto al final del sprint 2.

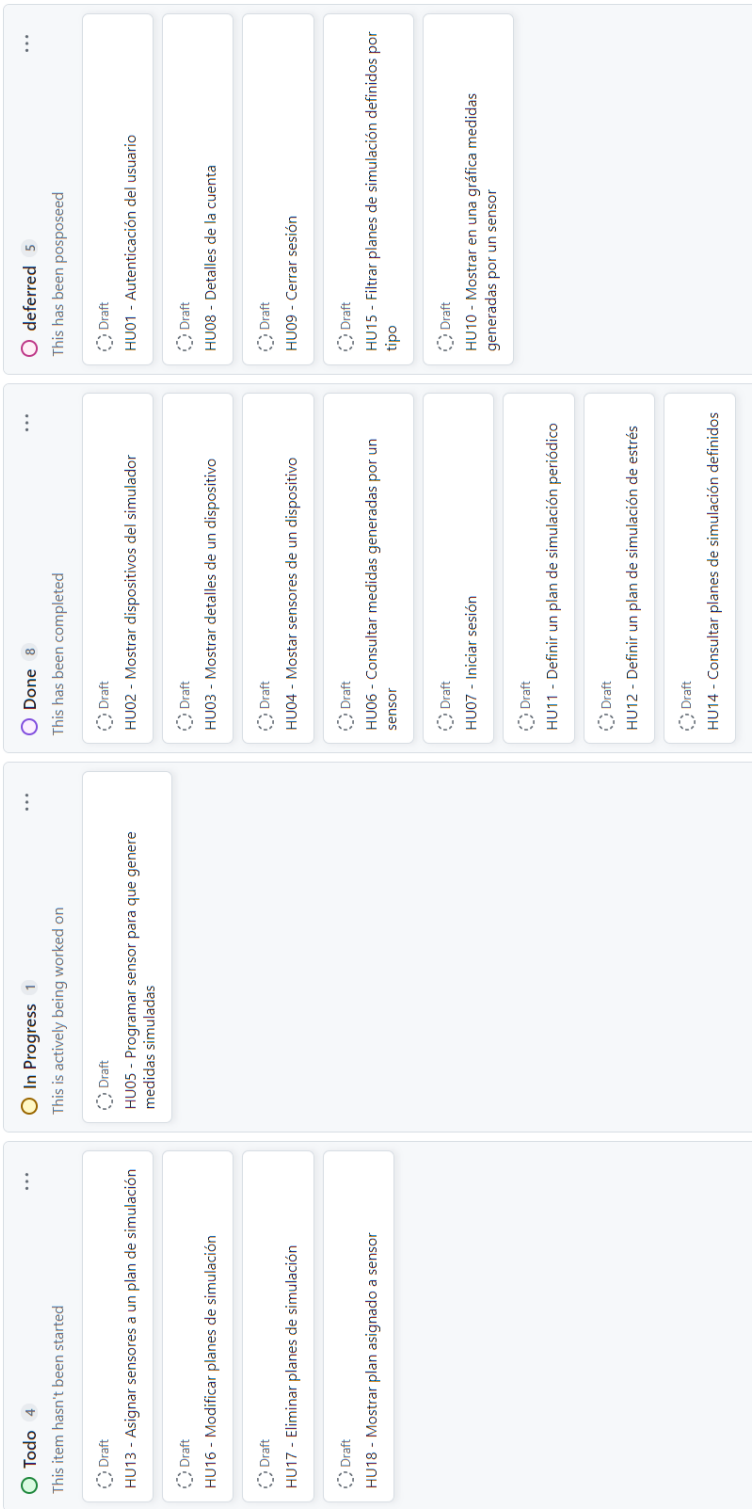



Figura 3.12: Kanban final del sprint 2.

## 3.3. Sprint 3

Tras haber analizado y definido como simular un sensor durante el sprint anterior, en este momento fue necesario implementar las historias restantes para que los planes de simulación estuviesen operativos.

### 3.3.1. Pila del producto

Para este sprint se eligió la historia de usuario HU13, que permitía asignar sensores a un plan, con el fin de completar el proceso de definición de los planes de simulación.

También se optó por trabajar en las historias de usuario de editar y eliminar planes, HU16 y HU17 respectivamente, con el objetivo de completar todas las funcionalidades CRUD (crear, leer, actualizar y borrar) para los planes de simulación.

Además, se escogió la historia de usuario HU18 recién añadida para mostrar el plan asignado a un sensor al ver sus detalles.

La pila del tercer sprint se muestra en la figura 3.13.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU05 - Programar sensor para que genere medidas simuladas | In Progress | simulation | very high |
| HU13 - Asignar sensores a un plan de simulación | Todo | simulation | very high |
| HU16 - Modificar planes de simulación | Todo | simulation | high |
| HU17 - Eliminar planes de simulación | Todo | simulation | medium |
| HU18 - Mostrar plan asignado a sensor | Todo | sensors | low |

Figura 3.13: Pila del sprint 3.

### 3.3.2. Desarrollo del sprint

La primera historia de usuario a implementar durante este sprint fue la HU13, que implicaba asignar sensores a un plan de simulación. Esto completó el tercer y último paso para crear planes de simulación.

Además, para simplificar la tarea de asignar un sensor, se incorporó una función de búsqueda que permitía filtrar la tabla de sensores disponibles en el simulador. A diferencia de un buscador convencional, este no tenía un botón para iniciar la búsqueda, sino que utilizaba un *debouncer*[6]. Con este mecanismo, la tabla se filtraba cada vez que el usuario dejaba de escribir, teniendo así un comportamiento más cómodo y realizando el filtrado en tiempo real. De esta manera, el usuario podía ver la lista acotada en base al texto introducido, facilitando la localización del sensor deseado. Adicionalmente, al asignar o desasignar un sensor a un plan se requería tener un control que comprobase las posibles asignaciones previas del sensor a otro plan y que no

quedase en un estado inconsistente.

Posteriormente, para permitir la edición (HU16), se tuvo que añadir una nueva funcionalidad a la tabla que mostrara el botón de editar cuando se le indicara y que al pulsarlo se redirigiese a la URL (Uniform Resource Locator) proporcionada. Aunque el proceso de edición tenía los mismos pasos que la creación de un nuevo plan de simulación y se reutilizaban los componentes, se tuvo que añadir la nueva lógica correspondiente al servicio encargado de mantener el estado y a cada uno de los componentes para que se comportasen de manera diferente dependiendo de si se estaba añadiendo o editando. Además, para este proceso de edición creí conveniente que no se permitiera escoger el tipo del plan, ya que parecía que lo coherente fuera que si se necesitaba cambiar el tipo se crease uno nuevo, puesto que la naturaleza de ambos tipos era realmente diferente. De este modo, se iniciaba el proceso directamente en el segundo paso de definición de sus características.

Por otro lado, y pese a que no era un historia de usuario, se añadió un último paso a ambos procesos, donde se mostraba un resumen del plan creado y los sensores asignados junto a un botón para confirmar la creación.

Tras establecer el proceso para definir nuevos planes de simulación y editarlos, se debía implementar la funcionalidad para gestionar los schedulers definidos por los planes. Para ello, fue necesario ajustar la parte backend de la aplicación. Esta implementación debía generar medidas para los sensores de los planes, programar este proceso para que se repitiera de acuerdo a su frecuencia de muestreo y, además, se debía reiniciar el programador al modificar un plan, para que se ejecutaran de acuerdo a la nueva definición.

Continuando con toda esta parte de los planes de simulación, se mejoró el componente tabla, en esta ocasión para que también mostrara el botón de eliminación de un plan (HU17), incluyéndose un parámetro para indicar la función a ejecutar. Al pulsar este botón se tenía que mostrar un mensaje de confirmación de la acción. Por último, este proceso también debía borrar las medidas generadas, desasignar todos sus sensores y cancelar la ejecución de su scheduler. Al eliminar las medidas se insertaban en una tabla de historial.

Finalmente, se añadió a la página de detalles de un sensor un apartado para que, a parte de mostrar sus características y las medidas simuladas, mostrase también información relativa al plan de simulación asignado y sus particularidades.

### 3.3.3. Detalles de la implementación

A continuación se detalla como se abordaron las distintas historias de usuario de este sprint y se hace una valoración de los resultados obtenidos:

- **HU13 - Gestión de Planes de Simulación:** esta historia que consistía en asignar sensores a un plan de simulación al crearlo, gracias a la tabla reutilizable y a los métodos existentes para recuperar los sensores, se pudo realizar con éxito en un tiempo menor al esperado. Cabe que recordar que para implementar funcionalidades anteriores ya se tuvieron que listar algunos sensores, con lo que se estandarizaron métodos.

- **HU18 - Gestión de Sensores:** esta funcionalidad que permitía al usuario ver el plan periódico asignado a un sensor, ayudaba a comprender rápidamente como se generan los datos para ese sensor y permitía comprobar si la asignación se había hecho correctamente.
- **HU16 - Gestión de Planes de Simulación:** para esta historia, se había implementado una funcionalidad para editar un plan de simulación. Sin embargo, durante la demostración al final del sprint, el supervisor de las prácticas creyó más conveniente que el proceso de edición fuera igual al de creación de un plan, sin saltarse el primer paso. Por eso, esta historia se devolvió al product backlog para ser modificada en otro sprint.
- **HU17 - Gestión de Planes de Simulación:** se implementó la funcionalidad para borrar un plan de simulación. De este modo, el usuario era capaz de borrar un plan de simulación si ya no lo necesita.

En este sprint se lograron terminar todas las historias de usuario, excepto el pequeño ajuste en la HU16.

La pila del producto y el Kanban al final de este sprint se pueden ver en las figuras 3.14 y 3.15.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | deferred | auth | very high |
| HU08 - Detalles de la cuenta | deferred | auth | Very low |
| HU09 - Cerrar sesión | deferred | auth | Very low |
| HU10 - Mostrar en una gráfica medidas generadas por un sensor | deferred | generated data | medium |
| HU15 - Filtrar planes de simulación definidos por tipo | deferred | simulation | Very low |
| HU16 - Modificar planes de simulación | In Progress | simulation | high |

Figura 3.14: Pila del producto al final del sprint 3.

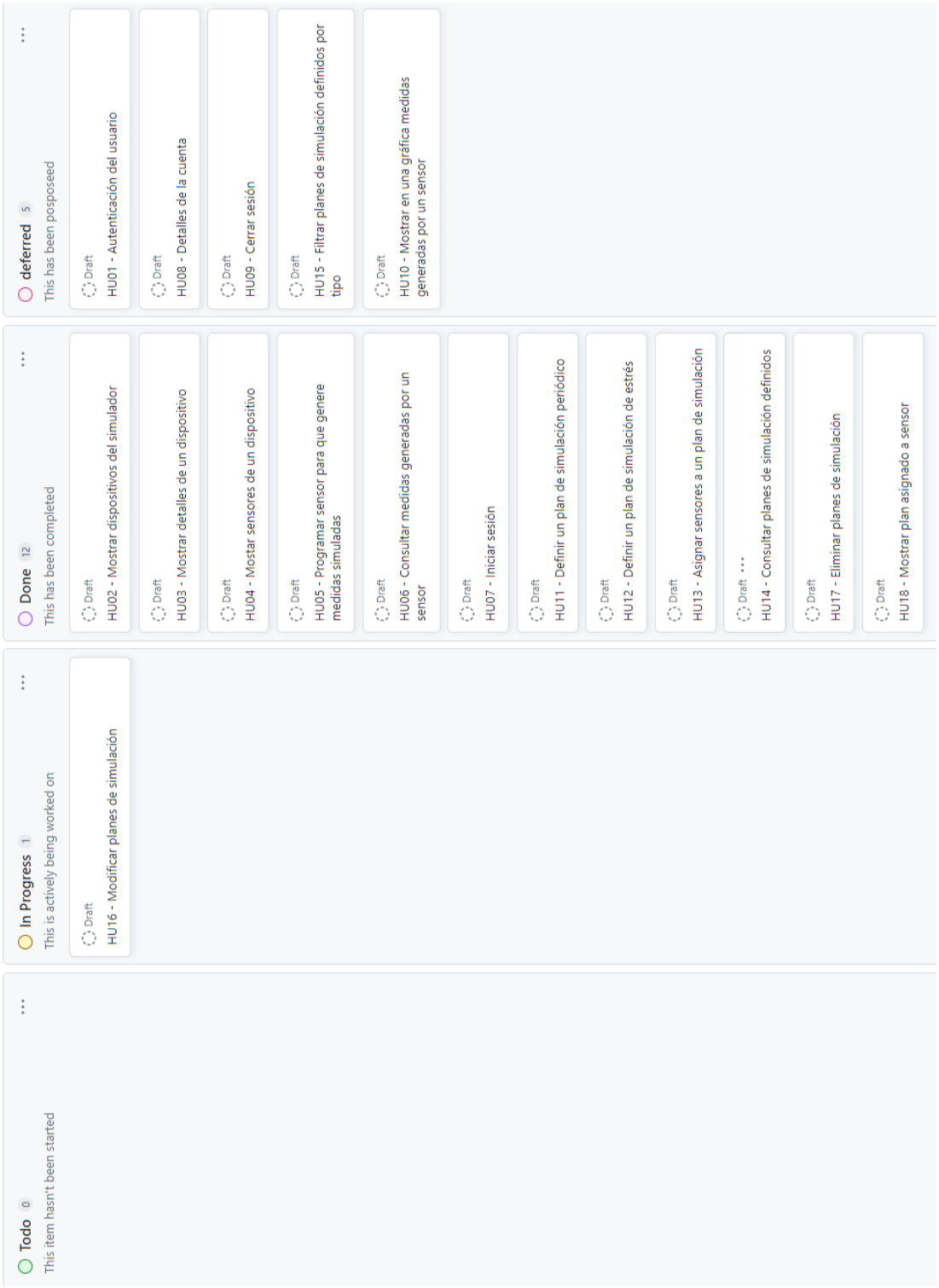



Figura 3.15: Kanban al final del sprint 3.

## 3.4. Sprint 4

Para este sprint, teniendo ya medidas totalmente aleatorias cada cierto intervalo de tiempo especificable, se consideró relevante poder definir como se comportan las medidas simuladas. Para ello, se hizo un análisis y diseño sobre este aspecto.

### 3.4.1. ¿Cómo simular un sensor?

Como se había mencionado anteriormente, para simular el comportamiento de un sensor real, era imprescindible considerar la frecuencia con la que se deseaba tomar una muestra del entorno. Pero también era crucial el proceso de obtención de esa muestra. Esto implicaba imitar y replicar el procedimiento mediante el cual un sensor adquiere, procesa y transmite la información. De esta forma, se aseguraba que los datos simulados reflejasen de manera precisa lo que el sensor habría experimentado bajo idénticas condiciones en un entorno real.

### 3.4.2. ¿Cómo el simulador da respuesta a los requisitos?

Inicialmente, se consideró establecer un intervalo y una dirección para el flujo de datos de cada sensor (ASC, DESC, RND). Sin embargo, esta metodología comportaba tener mucha información duplicada e iba a ser incómodo de usar. Por consiguiente, se decidió adoptar un enfoque similar a la de los planes de simulación. En este caso, se denominaron *planes de comportamiento*.

Estos planes se encargarían de establecer las pautas del comportamiento de los datos simulados siguiendo un modelo específico. Este patrón reflejaría las condiciones de un entorno específico.

Además, se identificó la necesidad de definir dos comportamientos distintos al llegar a la cota del intervalo: continuar emitiendo el valor tope, o reiniciar y volver a empezar desde el extremo inferior o superior del intervalo, según la dirección del flujo de datos.

Por otro lado, como ya se había indicado, estos planes permitirían tres configuraciones principales para definir las mediciones del sensor:

1. **Comportamiento Aleatorio**: en este modo, las mediciones del sensor fluctuarían de manera aleatoria, lo que puede ser útil para simular entornos con condiciones variables e impredecibles.
2. **Comportamiento Ascendente**: en este modo, las mediciones del sensor seguirían una secuencia ascendente, donde cada medición es igual o superior a la anterior, pero nunca inferior. Estas mediciones se realizarían dentro de un rango determinado por el usuario. Además, la variación entre cada una de las mediciones sería un número aleatorio pero limitado para evitar variaciones demasiado grandes. Cuando se alcanzara la cota superior, el plan podría configurarse para continuar simulando el valor del límite o para que volviese a empezar desde el valor mínimo.

3. **Comportamiento Descendente**: en este modo, las mediciones del sensor seguirían una secuencia descendente, donde cada medición sería igual o inferior a la anterior, pero nunca superior. Cuando se alcanzase la cota inferior, el plan podría configurarse para que continuara simulando el valor del límite o para que volviese a empezar desde el valor máximo.

### 3.4.3. Pila del producto

Como se ha visto, resultaba imprescindible definir una categoría adicional de planes, distinta a los de simulación, lo cual requirió añadir las historias correspondientes al product backlog.

En la figura 3.16, se puede ver la pila del sprint con las historias de usuario asignadas y las recién añadidas.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU16 - Modificar planes de simulación | In Progress | simulation | high |
| HU10 - Mostrar en una gráfica medidas generadas por un sensor | In Progress | generated data | medium |
| HU19 - Definir plan de comportamiento aleatorio | Todo | behaviour | high |
| HU20 - Definir plan de comfporttamiento ascendente | Todo | behaviour | high |
| HU21 - Definir plan de comportamiento descendente | Todo | behaviour | high |
| HU22 - Asignar sensores a un plan de comportamiento | Todo | behaviour | high |
| HU23 - Modificar un plan de comportamiento | Todo | behaviour | medium |
| HU24 - Eliminar un plan de comportamiento | Todo | behaviour | low |

Figura 3.16: Pila del sprint 4.

### 3.4.4. Desarrollo del sprint

Al comienzo de este sprint, se retomó la historia HU16, que se había abordado en el sprint anterior, para modificarla según las indicaciones realizadas por parte del supervisor de prácticas.

Además, para mejorar la comprensión de las tendencias de las medidas, decidí implementar un gráfico para las métricas, una tarea aplazada desde el primer sprint, de modo que se pudieran visualizar tanto en formato tabla como en un gráfico, facilitando la interpretación de las medidas.

En el sprint 1 se desarrolló una gráfica que mostraba todas las medidas del simulador; el objetivo actual era adaptarlo para mostrar únicamente las medidas de un sensor específico. Observando la utilidad del componente generalizado de la tabla, se procedió a parametrizarlo lo máximo posible. Por último, para poder cambiar entre la vista de tabla y la de gráfica, se creó un *toggle button* que permitiese alternar fácilmente de un modo a otro.

En la figura 3.17 se muestra, a modo de ejemplo, las medidas de un sensor en este nuevo componente gráfico y el *toogle button* para cambiar el modo de ver las medidas.

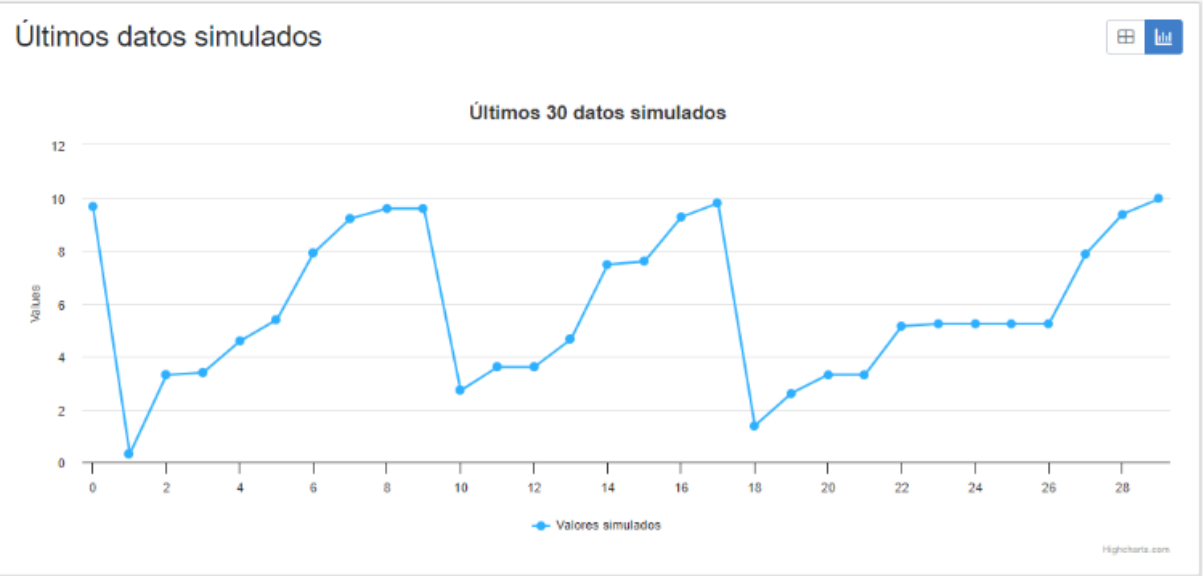



Figura 3.17: Gráfica con las medidas de un sensor junto con el *toogle button*.

Como hito principal, se tuvieron que implementar las historias para gestionar los nuevos planes de comportamiento. Primero, se modificó el diseño de la base de datos, para permitir almacenar la información relativa a estos planes, añadiendo nuevos scripts de Flyway al proyecto. El resultado final tras los reajustes se muestra en la figura 3.18.

A continuación, se creó una nueva sección en el menú lateral llamada *Comportamientos* para permitir acceder a los planes de comportamiento. Al entrar a esta sección, se presentaban estos planes de manera análoga a como se listaban los planes de simulación.

A esta página se incorporó un botón para la creación de planes de comportamiento. Este proceso, al igual que el de los planes de simulación, también requería de varios pasos. Primero se debía definir el flujo de sus datos (aleatorio, ascendente o descendente) junto a las cotas (inferior y superior) del intervalo de generación de medidas. Además, para este primer paso consideré conveniente mostrar una gráfica de previsualización del comportamiento generado para que el usuario pudiese comprender más fácilmente el plan que está definiendo. Para implementar esta gráfica se pudo usar el componente reutilizable diseñado anteriormente y se crearon unas secuencias de valores de ejemplo para cada uno de los flujos disponibles. Continuando con el proceso, en el siguiente paso se asignaban los sensores pertinentes al plan, con el fin de que sus medidas tuviesen ese comportamiento. Por último, se presentaba una página de confirmación, mostrando un resumen de las particularidades del plan junto a los sensores asignados.

Posteriormente, se aprovecharon las mejoras realizadas en sprints anteriores al componente de la tabla para implementar las funciones de edición y eliminación, siguiendo el mismo protocolo establecido. El proceso de edición, en esta ocasión, era idéntico al de creación desde el principio.

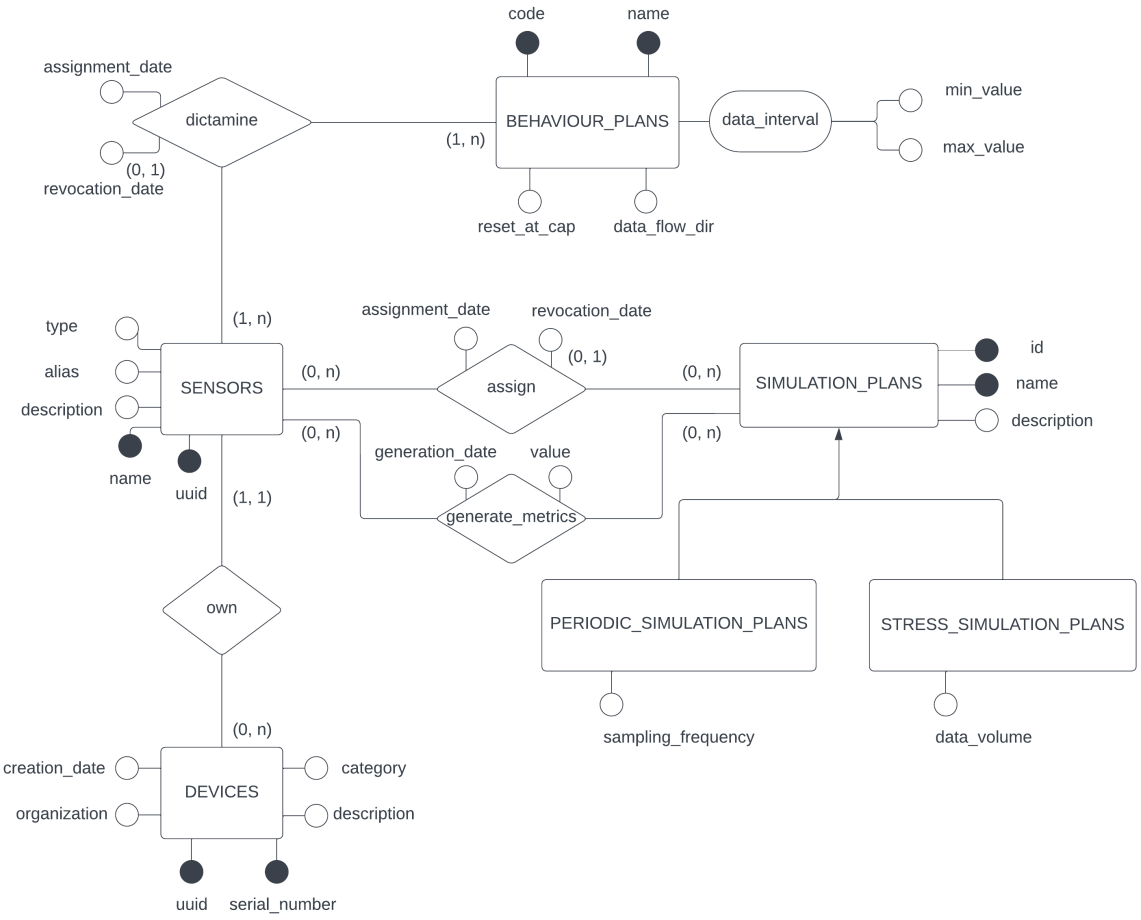



Figura 3.18: Diseño conceptual de la base de datos tras las modificaciones del sprint 4.

### 3.4.5. Detalles de la implementación

A continuación, se detallan las historias de usuario y los resultados obtenidos:

- **HU16 - Gestión de Planes de Simulación:** se modificó esta historia según las indicaciones del supervisor de prácticas. De este modo, el proceso de edición era idéntico al de creación de un plan, lo que mejoraba la coherencia y la usabilidad de la aplicación.
- **HU19, HU20, HU21, HU22, HU23 y HU24 - Gestión de Planes de Comportamiento:** estas historias, que se añadieron al backlog en este sprint, implicaban la creación, modificación y eliminación de planes de comportamiento que definían como se simulan las medidas de los sensores. Gracias a la experiencia adquirida del anterior proceso de gestión de planes de simulación, se pudieron completar todas estas historias en este sprint.

- **HU10 - Visualización de Medidas en un Gráfico:** como en este sprint se definieron comportamientos para las medidas, se decidió retomar la HU10, que quedó aplazada en el primer sprint, para poder visualizar fácilmente las tendencias de las medidas.

Este sprint, aunque a priori parecía muy complicado, gracias a la experiencia ya adquirida hasta este punto, pudo ser completado con éxito.

La pila del producto y Kanban resultante se muestran en la figura 3.19 y 3.20 respectivamente. En este último, se incluyen únicamente las transacciones más recientes, debido a las limitaciones de espacio en la captura de pantalla. Además, las transacciones anteriores ya han sido documentadas en los informes de los sprints previos.

| | Title | Status | Scope | Priority |
|---|---|---|---|---|
| 1 | HU01 - Autenticación del usuario | deferred | auth | very high |
| 2 | HU08 - Detalles de la cuenta | deferred | auth | Very low |
| 3 | HU09 - Cerrar sesión | deferred | auth | Very low |
| 4 | HU15 - Filtrar planes de simulación definidos por tipo | deferred | simulation | Very low |

Figura 3.19: Estado de la pila del producto al final del sprint 4.


Figura 3.20: Estado del Kanban al final del sprint 4.

## 3.5. Sprint 5

Llegados a este sprint, con la capacidad de simular medidas de los sensores en intervalos definidos y pudiendo especificar su comportamiento, se consideró que el paso siguiente fuera la importación de dispositivos y sensores desde Control Platform, así como el almacenamiento de las medidas simuladas en la base de datos central.

Para llevar a cabo esta nueva funcionalidad, le propuse al supervisor de prácticas la idea de un proceso interactivo que tuviese varios pasos. Esto permitiría a los usuarios seleccionar los dispositivos y sensores a importar, asignarles planes de simulación y comportamiento, y finalizar con un resumen de la importación con el respectivo botón de confirmación para hacer efectiva la acción. Por último, para diferenciar los dispositivos reales de los simulados, se acordó que solo se podrían importar aquellos dispositivos cuyo identificador que empezara por *SIM*.

### 3.5.1. Pila del producto

Durante la planificación de este sprint, se añadieron a la pila del producto todas las nuevas historias relativas al proceso de importar.

Por otro lado, se observó que, a pesar de tener implementada toda la lógica de los planes de simulación de estrés, no existía ningún botón que permitiera su ejecución. Por tanto, se incorporó una historia de usuario para cubrir este aspecto.

Además, se consideró que sería beneficioso poder ejecutar la simulación de estrés para un sensor sin necesidad de asignarlo previamente, evitando así interrumpir su plan de simulación periódico que simulaba su funcionamiento normal. De este modo, en la página de detalles se habilitó una opción para simular una cantidad específica de medidas, proporcionadas por un plan de estrés o introducidas directamente por el usuario. Por lo tanto, se incorporaron dos historias de usuario adicionales relacionadas con estas nuevas formas de simular medidas para un sensor.

En definitiva, la pila del producto con estas historias nuevas añadidas y la pila concreta de este sprint, se pueden ver en las figuras 3.21 y 3.22 respectivamente.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | deferred | auth | very high |
| HU08 - Detalles de la cuenta | deferred | auth | Very low |
| HU09 - Cerrar sesión | deferred | auth | Very low |
| HU15 - Filtrar planes de simulación definidos por tipo | deferred | simulation | Very low |
| HU25 - Consultar plan de comportamiento asignado a sensor | Todo | sensors | low |
| HU26 - Ejecutar plan de estrés concreto | Todo | simulation | medium |
| HU27 - Ejecutar plan de estrés para un sensor una vez sin asignación | Todo | simulation | medium |
| HU28 - Simular cantidad de datos específica | Todo | simulation | medium |
| HU29 - Guardar medidas simuladas en base de datos central | Todo | generated data | very high |
| HU30 - Importar dispositivos de base de datos central | Todo | devices | very high |
| HU31 - Importar sensores de un dispositivo | Todo | sensors | high |
| HU32 - Filtrar medidas simuladas de un sensor por plan de simulación | Todo | generated data | low |
| HU33 - Asignar planes a las importaciones | Todo | simulation | high |

Figura 3.21: Pila del producto al principio del sprint 5.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU25 - Consultar plan de comportamiento asignado a sensor | Todo | sensors | low |
| HU26 - Ejecutar plan de estrés concreto | Todo | simulation | medium |
| HU27 - Ejecutar plan de estrés para un sensor una vez sin asignación | Todo | simulation | medium |
| HU28 - Simular cantidad de datos específica | Todo | simulation | medium |
| HU29 - Guardar medidas simuladas en base de datos central | Todo | generated data | very high |
| HU30 - Importar dispositivos de base de datos central | Todo | devices | very high |

Figura 3.22: Pila del sprint 5.

### 3.5.2. Desarrollo del sprint

La primera historia de usuario que se abordó durante este sprint fue la HU25, relativa a mostrar los detalles del plan de comportamiento asignado a un sensor. Para ello, aunque al principio parecía relativamente sencillo ya que era similar a la de mostrar los detalles del plan de simulación, fue un poco más complejo de lo esperado. Esto se debió a que se quería evitar el uso de una barra de desplazamiento y poder ver todos los detalles del sensor de un solo vistazo. Para ello, se tuvo que implementar un *toogle button* que permitiese alternar entre mostrar los detalles de un tipo de plan u otro. En la figura 3.23 se pueden ver los detalles de un plan de comportamiento junto con el *toggle button*.

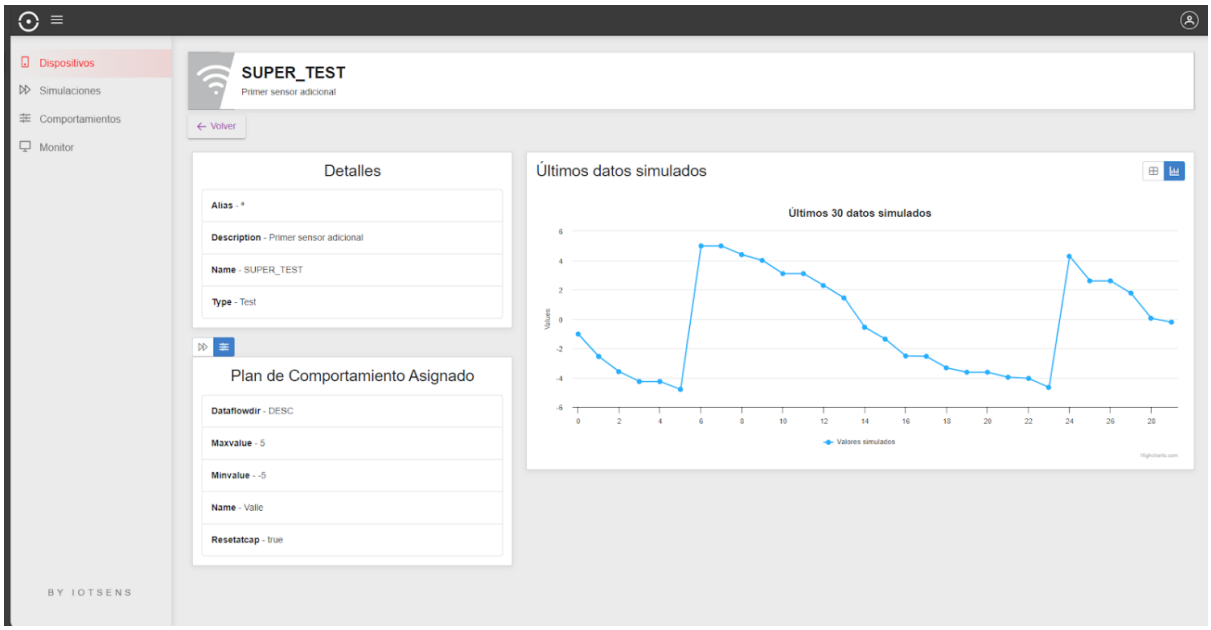



Figura 3.23: Página de detalles de un sensor con su plan de comportamiento.

Posteriormente, para habilitar la ejecución de un plan de estrés (HU26), se adaptó el componente generalizado de la tabla para incluir un botón de ejecutar únicamente en caso de que mostrara un plan de estrés. Una vez añadido el botón, fue suficiente con realizar una llamada al endpoint del controlador, dado que el resto de la implementación ya estaba desarrollada.

Antes de empezar a explicar las siguientes historias del sprint, y con el fin de entender porque se definieron, cabe recordar algunos aspectos de los planes del simulador. Hasta ese momento habían definidos dos tipos de planes. A su vez, los planes de simulación tenían dos tipos, como puede observarse en la figura 3.24.

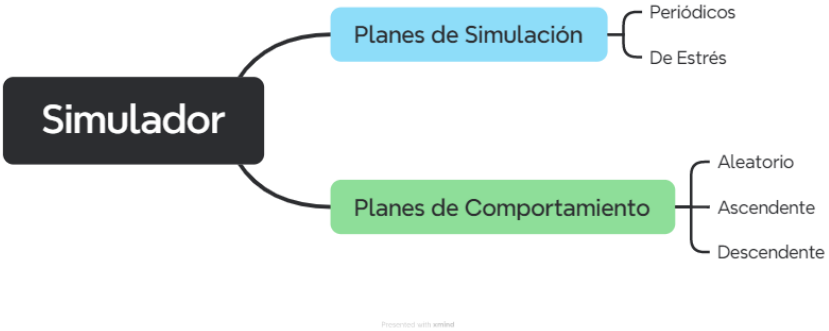



Figura 3.24: Tipos de planes del simulador.

Los planes de simulación periódicos, definían la frecuencia de muestreo, mientras que los planes de simulación de estrés, permitían establecer el volumen de medidas a generar. Los primeros emulaban el comportamiento normal de un sensor, los otros servían para pruebas de carga.

Tal y como estaba definido el modelo, un sensor sólo podía tener asignado un plan de simulación al mismo tiempo. Por tanto, si se requiriesen efectuar pruebas de estrés, al asignar el plan de simulación de estrés a los sensores, se perdería el plan de simulación periódico que tenían asignado hasta ese momento. Con el fin de evitar esta situación, y tras valorar diferentes alternativas, se consideró más adecuada la opción de implementar un menú para poder ejecutar un plan de estrés sin necesidad de asignarlo al sensor, manteniendo de este modo su plan periódico. Añadir que en el plan de estrés se definía el volumen de datos, por tanto al seleccionar el plan de estrés a ejecutar, se iban a generar para ese sensor la cantidad de medidas especificadas por dicha definición.

Llegados a este punto, era posible dar un paso más y permitir simular una cantidad de datos especificable en vez de restringirse a que existiese un plan de estrés con ese volumen de datos para seleccionarlo. Por consiguiente, se implementó también la posibilidad de especificar la cantidad de datos. En resumen, se añadió un menú con en el que o bien, se escogía el plan de estrés a ejecutar y, por tanto, se simulaban la cantidad de medidas definidas por el plan, o bien se especificaba esa cantidad directamente desde el *input*.

Para implementar este menú que permitiese una simulación a la carta (HU27 y HU28), primero se añadió un botón en la página de detalles del sensor que permitía abrir el menú lateral para personalizar la simulación. Para ello, en este menú se incluyó un desplegable con los planes de simulación de estrés creados, ofreciendo al usuario la posibilidad de escoger el que considerara oportuno (HU27). Además, en dicho menú también se añadió un *toggle button* para cambiar de modo, mostrando un *input* para introducir la cantidad de medidas que se deseaban generar (HU28).

En lo concerniente a la implementación del backend, dado que la HU26 ya estaba desarrollada, el proceso seguido en la HU27 fue similar, pero en este caso solo se aplicaba para un sensor específico y sin la necesidad de estar asignado al plan. Para la HU28, en la que se permitía poder simular cantidades específicas sin que se seleccionase ningún plan, se creó un plan de

estrés interno transparente al usuario, con un identificador no generable automáticamente. Una vez creado este plan comodín, el resto fue equivalente a la HU27.

Una vez finalizadas estas historias de usuario, se abordó la comunicación con la base de datos central. El primer paso fue la creación de un dispositivo en Control Platform cuyo número de serie empezaba por *SIM* y que tenía sensores creados aleatoriamente. Se creó en el entorno de test para evitar interferencias en el funcionamiento normal de esta aplicación.

Tras este paso, y pese a no parecer ser el orden lógico, se decidió almacenar las medidas en su base de datos central antes de empezar a importar dispositivos y sensores. Por tanto, fue necesario insertar el dispositivo creado en Control Platform en el simulador de forma manual, ya que el proceso de importación aún no estaba implementado.

Posteriormente, usando la API (Application Programming Interface) interna que la empresa tenía desarrollada, se tuvo que implementar un controlador con un endpoint para preparar la petición y hacer la llamada a esa API, con el fin de enviar las medidas para que fuesen almacenadas en la base de datos central. Tras conseguir almacenar las medidas en dicha base de datos, al acceder al dispositivo recién creado, ya eran visibles en Control Platform.

El siguiente paso fue importar los dispositivos y sensores al simulador. Para ello, primero se eliminó el dispositivo y sensores importados manualmente. Posteriormente, se añadió un nuevo endpoint al controlador anterior para, utilizando la API de la empresa, obtener los dispositivos y sensores. Esta API ya permitía aplicar filtros a los atributos de los dispositivos, por lo que, podía emplearse para obtener los dispositivos cuyo número de serie empezara por *SIM*. Una vez obtenidos, se listaron en el componente de la tabla reutilizable.

### 3.5.3. Detalles de la implementación

Recapitulando, en este sprint se implementaron las historias de usuario relacionadas con la importación de dispositivos y sensores desde Control Platform, la ejecución de planes de simulación de estrés sin asignación y la simulación personalizada de una cantidad de datos especificable.

Además, hubo que guardar las medidas simuladas en la base de datos central usando una API que la empresa tenía creada.

A continuación, se detallan las historias de usuario y los resultados obtenidos:

- **HU25 - Consultar el comportamiento de un sensor:** la implementación de esta historia fue similar a la de mostrar el plan de simulación asignado. Además, llegados a este punto, tras observar se había implementado una lista para los detalles de dispositivos, sensores y planes, se optó por desarrollar un componente generalizado. Este componente, recibía como entrada el objeto a mostrar y presentaba todos sus atributos en una lista con estilos.

- **HU26 - Ejecutar plan de simulación de estrés:** la realización de esta historia requirió la adición de un botón de ejecución a la lista de planes de simulación, en el caso de que el plan fuera de estrés. La implementación del backend era parecida a la de los planes periódicos, pero menos compleja porque no requería el uso de schedulers.
- **HU27 y HU28 - Simulación a la carta:** para la implementación de estas historias de usuario se creó un plan de estrés interno con un código especial. Puesto que los códigos de los planes se generaban automáticamente siguiendo un patrón, se creó un código fuera de dicho patrón.
- **HU29 - Guardar medidas simuladas en base de datos central:** a pesar de ser una de las historias más importantes y difíciles del simulador, su implementación resultó no ser tan compleja como parecía gracias a una API que la empresa tenía desarrollada. No obstante, fue necesario aprender qué llamadas hacer a esta API y personalizarla para adaptarla a los requerimientos de las entidades del simulador.
- **HU30 - Importar dispositivos**: para esta historia, se utilizó nuevamente la API, filtrando el número de serie por aquellos que comienzan por *SIM*. Además, fue necesario crear un nuevo elemento en el menú lateral y una nueva ruta en la aplicación web. Gracias a la existencia de todas las otras rutas creadas y una buena gestión de dependencias, solo se requirieron unas pequeñas modificaciones. Finalmente, se implementó una restricción para evitar la importación de dispositivos ya importados.

La pila del producto y el Kanban al final del sprint se pueden ver en las figuras 3.25 y 3.26.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | deferred | auth | very high |
| HU08 - Detalles de la cuenta | deferred | auth | Very low |
| HU09 - Cerrar sesión | deferred | auth | Very low |
| HU15 - Filtrar planes de simulación definidos por tipo | deferred | simulation | Very low |
| HU31 - Importar sensores de un dispositivo | Todo | sensors | high |
| HU32 - Filtrar medidas simuladas de un sensor por plan de simulación | Todo | generated data | low |
| HU33 - Asignar planes a las importaciones | Todo | simulation | high |

Figura 3.25: Pila del producto al final del sprint 5.

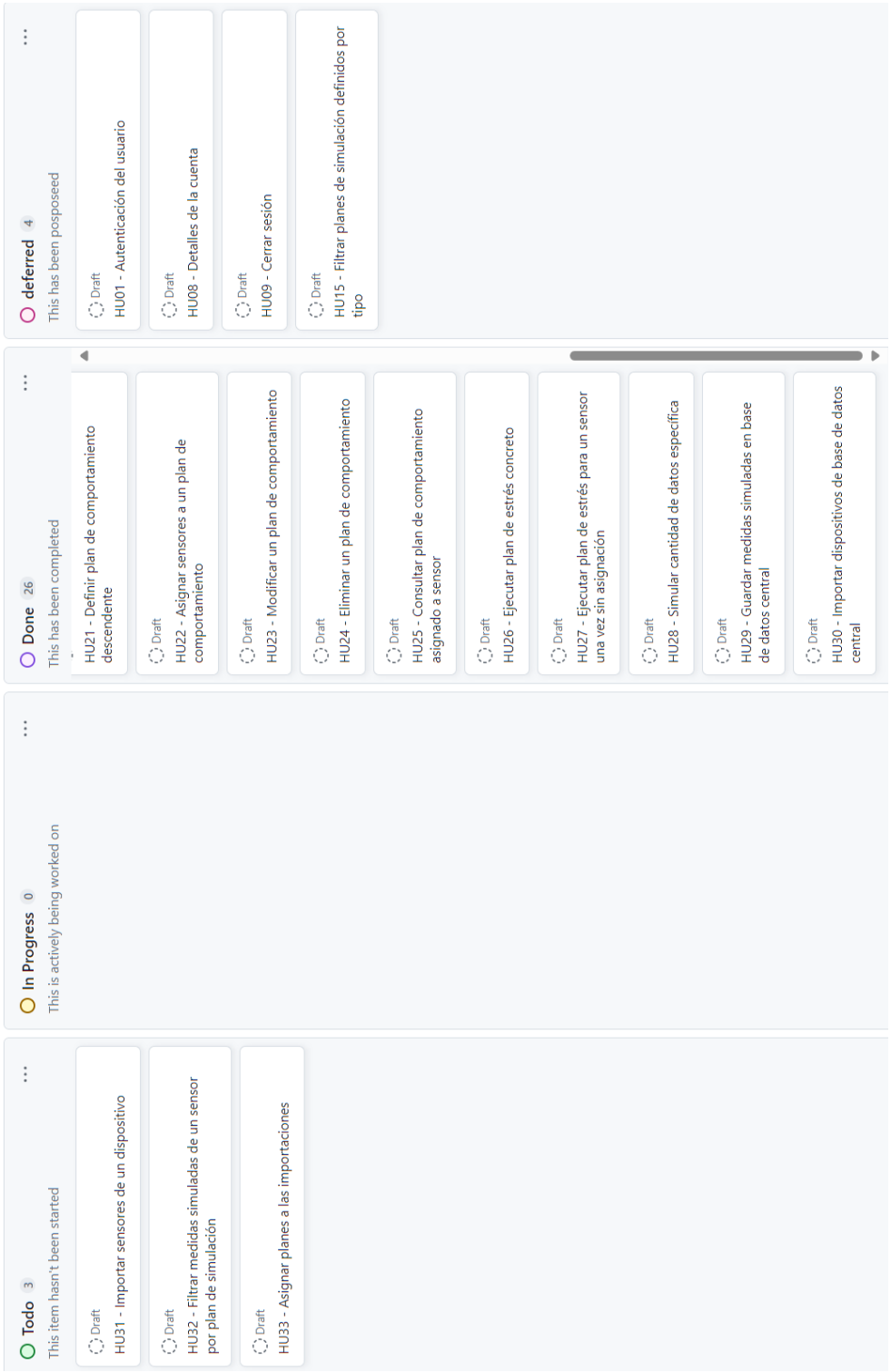



Figura 3.26: Kanban al final del sprint 5.

## 3.6. Sprint 6

El objetivo de este sprint era terminar el proceso de importación de dispositivos y sensores desde Control Platform.

### 3.6.1. Pila del producto

Para completar la importación, tras conseguir listar los dispositivos aptos para este proceso en el sprint anterior, quedaban por implementar las historias de usuario HU31 y HU33.

La HU33 que reflejaba la necesidad de poder asignar planes a los sensores al importarlos, dado que un sensor requería al menos tener asignado un plan de simulación para que genere medidas, era demasiado general. Por ello, se desglosó en varias historias más pequeñas, concretándose más las especificaciones de como se quería permitir realizar este paso de la importación. De este modo, se convirtió en una épica.

El objetivo de estas nuevas historias de usuario originadas a partir de la HU33, fue, detallar como poder asignar planes de simulación y de comportamiento a cada uno de los sensores a importar. Ya fuese, individualmente o a nivel de dispositivo, permitiendo asignar el mismo plan a todos sus sensores.

El estado de la pila del producto con estas modificaciones y las historias de usuario asignadas para este sprint se pueden ver en las figuras 3.27 y 3.28.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | deferred | auth | very high |
| HU08 - Detalles de la cuenta | deferred | auth | Very low |
| HU09 - Cerrar sesión | deferred | auth | Very low |
| HU15 - Filtrar planes de simulación definidos por tipo | deferred | simulation | Very low |
| HU31 - Importar sensores de un dispositivo | Todo | sensors | high |
| HU33 - Asignar planes a las importaciones | Todo | simulation | high |
| HU34 - Asignar plan de simulación a un dispositivo al importar | Todo | simulation | high |
| HU35 - Asignar un plan de simulación a un sensor al importar | Todo | simulation | high |
| HU36 - Asignar un plan de comportamiento a un dispositivo al importar | Todo | simulation | high |
| HU37 - Asignar un plan de comportamiento a un sensor al importar | Todo | simulation | high |
| HU32 - Filtrar medidas simuladas de un sensor por plan de simulación | Todo | generated data | low |

Figura 3.27: Estado de la pila del producto al inicio del sprint 6.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU31 - Importar sensores de un dispositivo | Todo | sensors | high |
| HU33 - Asignar planes a las importaciones | Todo | simulation | high |
| HU34 - Asignar plan de simulación a un dispositivo al importar | Todo | simulation | high |
| HU35 - Asignar un plan de simulación a un sensor al importar | Todo | simulation | high |
| HU36 - Asignar un plan de comportamiento a un dispositivo al importar | Todo | simulation | high |
| HU37 - Asignar un plan de comportamiento a un sensor al importar | Todo | simulation | high |

Figura 3.28: Pila del sprint 6.

### 3.6.2. Desarrollo del sprint

Todas estas historias de usuario, desde un punto de vista técnico no eran excesivamente complejas de implementar, puesto que iban a requerir otro proceso multipasos, como el implementado para los planes de simulación y de comportamiento. En este aspecto ya se tenia experiencia, sin embargo sí que suponían un gran reto desde el punto de vista de la usabilidad.

El proceso de importar era posiblemente el primer contacto que iba a tener un usuario con el simulador y, además, estaba relacionado con todos los conceptos del mismo. A parte, también se necesitaba que fuera práctico. De modo que si se importaban muchos dispositivos, no fuera tedioso seleccionar los sensores a importar y hacer todas las asignaciones pertinentes. Por ello, para esta parte de asignaciones, definí una historia de usuario a nivel de sensor y otra a nivel de dispositivo.

Para implementar la primera historia de usuario, la HU31, decidí, con el fin de mejorar la usabilidad, mostrar los dispositivos seleccionados a importar en una tabla donde cada fila sería un dispositivo. Además, en cada una de ellas se añadiría un botón que permitiera importar todos sus sensores con un solo clic. De todos modos, también se permitía pulsar sobre un dispositivo para poder hacer una selección personalizada de los sensores a importar.

Posteriormente hubo que diseñar y desarrollar las historias de usuario relativas a asignar planes de simulación a los sensores a importar (HU34 y HU35). Para estas historias, tras pensar en una opción similar a la desarrollada para los sensores, al final, opté nuevamente por mostrar la lista de dispositivos a importar en una tabla con un botón para hacer la asignación del plan a nivel de dispositivo. A su vez, al clicar sobre un dispositivo en concreto, se permitía hacer la asignación de los planes a nivel de sus sensores. En esta ocasión, implementé un drag&drop para que el usuario pudiera repartir los sensores o el dispositivo entre los diferentes planes y poder tener una visión global del estado de las asignaciones en un solo vistazo. En la figura 3.29 se puede ver, en fase casi definitiva, esta nueva funcionalidad de arrastrar y soltar.

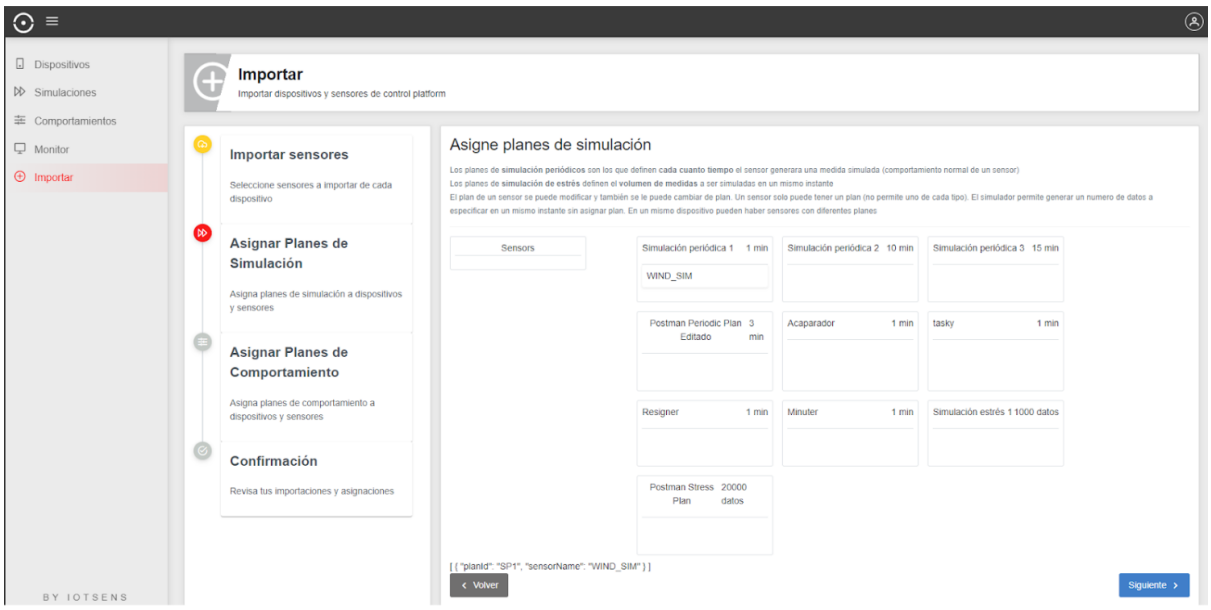



Figura 3.29: Ventana de asignación de planes de simulación al importar (en fase de desarrollo).

Adicionalmente, para las historias de usuario relativas a asignar los planes de comportamiento (HU36 y HU37), se implementó de una forma equivalente a la de asignar planes de simulación. La particularidad era que para estas se necesitaba mostrar información diferente sobre los planes, ya que eran de otro tipo. Así pues, aunque la base era la misma, se tuvieron que crear los pertinentes métodos de los servicios y generalizar los componentes para soportar los dos tipos de planes para el drag&drop.

Para terminar con el proceso de importación, pese a que no era explícitamente una historia de usuario y con el fin de facilitar este proceso al usuario, creí conveniente crear un paso final de confirmación, donde con un carrusel mostraba para cada dispositivo los sensores importados y las asignaciones, como se puede ver en la figura 3.30. Cada dispositivo se correspondía a una página distinta dentro del carrusel. Además, a cada paso del proceso de importación, incorporé una breve descripción de ayuda. Esta información adicional permitía al usuario comprender las acciones requeridas, ya que, posiblemente, este proceso constituiría su primer contacto con el simulador.

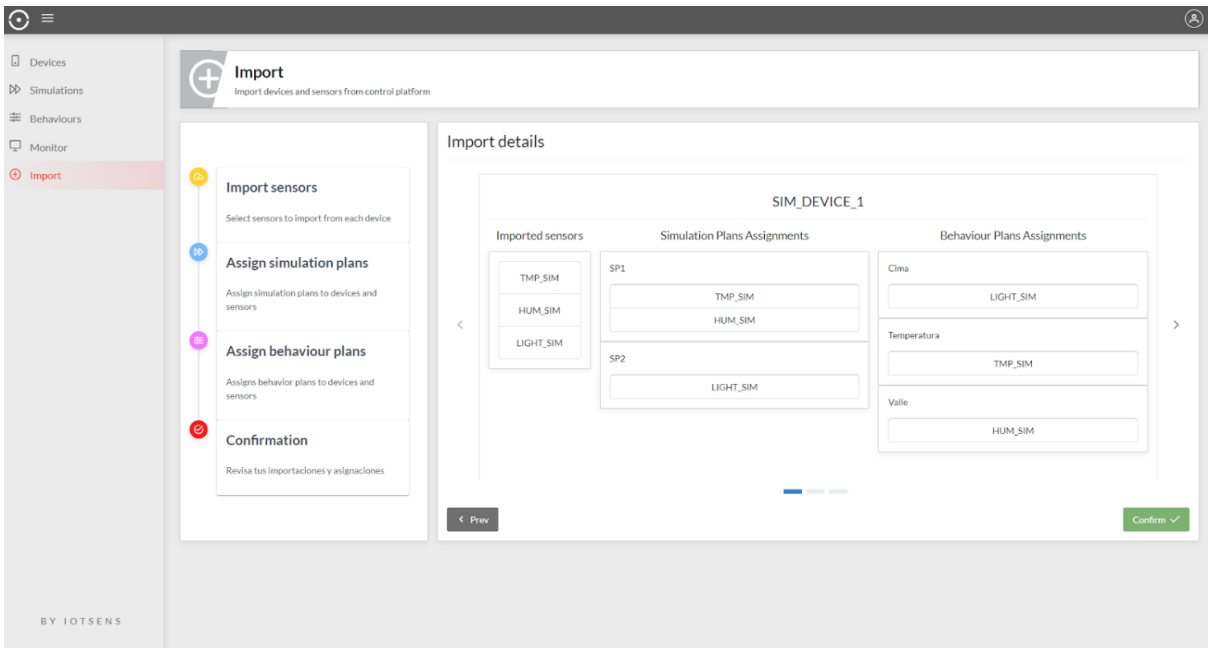



Figura 3.30: Ventana de confirmación del proceso de importación (con el simulador en inglés).

Por último, se considera importante mencionar, que llegado a este punto y con los conocimientos adquiridos, al final de este sprint, se refactorizaron los servicios y componentes de los procesos de crear y editar planes de simulación con el fin de adaptarlos a las nuevas metodologías aplicadas para procesos implementados posteriormente, como crear y editar planes de comportamiento o importar dispositivos y sensores de Control Platform.

### 3.6.3. Detalles de la implementación

A continuación se detallan las historias de usuario y los resultados logrados:

- **HU31 - Importar sensores:** la tabla implementada para esta historia de usuario también mostraba el número de sensores importados para cada dispositivo para facilitar al usuario un resumen de la situación, indicando incluso si el número de sensores importados era todos los que tenia el dispositivo.
- **HU34 y HU35 - Asignar planes de simulación:** en el drag&drop implementado para estas historias de usuario, se mostraba en una parte el dispositivo con sus sensores (arrastrables), mientras que en la otra zona se visualizan los planes de simulación existentes, en los que se puede soltar los sensores, junto a su nombre y la frecuencia de muestreo o volumen de datos.
- **HU36 y HU37 - Asignar planes de comportamiento:** este drag&drop, era prácticamente idéntico al anterior visualmente. El único cambio era que para los planes de comportamiento era necesario mostrar más detalles como el flujo de los datos, el intervalo, etc.

El estado de la pila del producto y el Kanban al final de este sprint se pueden ver en las figuras 3.31 y 3.32.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | deferred | auth | very high |
| HU08 - Detalles de la cuenta | deferred | auth | Very low |
| HU09 - Cerrar sesión | deferred | auth | Very low |
| HU15 - Filtrar planes de simulación definidos por tipo | deferred | simulation | Very low |
| HU32 - Filtrar medidas simuladas de un sensor por plan de simulación | Todo | generated data | low |

Figura 3.31: Pila del producto al final del sprint 6.

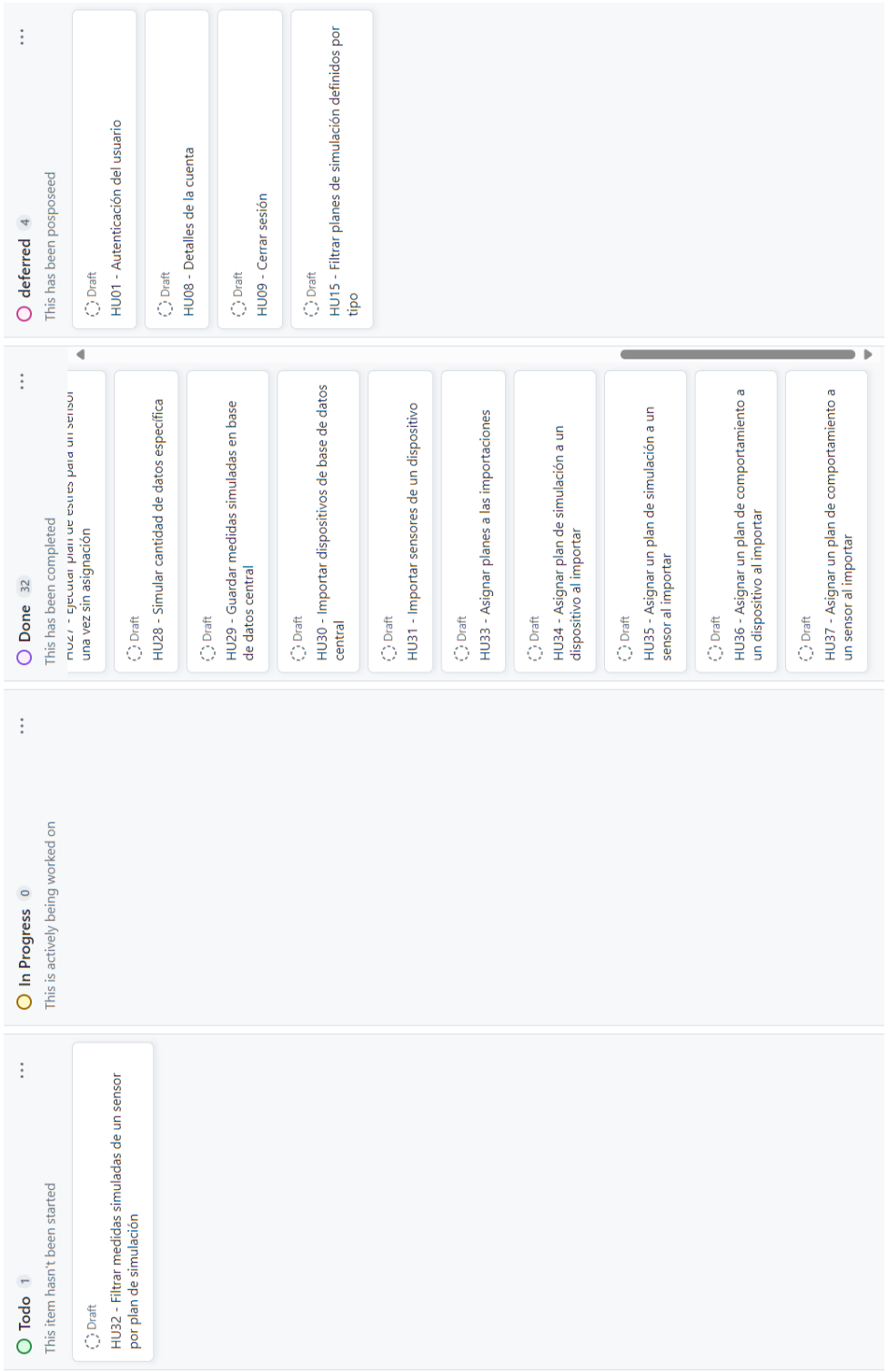



Figura 3.32: Kanban al final del sprint 6.

## 3.7. Sprint 7

Llegados a esta fase, se había completado la implementación de un simulador que cumplía ampliamente con las expectativas iniciales de la empresa cubriendo las necesidades expuestas a lo largo del desarrollo. Por tanto, las tareas pendientes se limitaban a finalizar aquellas que se pospusieron en sprints previos y a mejorar aspectos específicos de la usabilidad.

### 3.7.1. Pila del producto

Se omitió la tarea HU32 - Filtrar medidas simuladas de un sensor por plan de simulación, ya que la funcionalidad de envío de medidas simuladas a la base de datos central ya estaba operativa y accesible a través de la plataforma de Control Platform y esta plataforma permitía este filtrado, con lo que sería redundante.

Por otra parte, se introdujo el soporte plurilingüe al simulador, comenzando con la adición del idioma inglés. Esto implicaba la creación de una nueva historia de usuario para la internacionalización que permitiese la alternancia entre idiomas según las preferencias del usuario.

Finalmente, se planteó desplegar la aplicación en un entorno interno para facilitar su uso por parte de los desarrolladores de la empresa.

El estado actual de la pila del producto y las historias de usuario pendientes de implementar se presentan en las figuras 3.33 y 3.34.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | In Progress | auth | very high |
| HU08 - Detalles de la cuenta | In Progress | auth | Very low |
| HU09 - Cerrar sesión | In Progress | auth | Very low |
| HU15 - Filtrar planes de simulación definidos por tipo | In Progress | simulation | Very low |
| HU32 - Filtrar medidas simuladas de un sensor por plan de simulación | cancelled | generated data | low |
| HU38 - Adaptar simulador para ser multilingüe | Todo | usability | Very low |
| HU39 - Crear recursos para el español | Todo | usability | Very low |
| HU40 - Crear recursos para el inglés | Todo | usability | Very low |
| HU41 - Desplegar el simulador | Todo | | medium |

Figura 3.33: Estado de la pila del producto al inicio del sprint 7.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU01 - Autenticación del usuario | In Progress | auth | very high |
| HU08 - Detalles de la cuenta | In Progress | auth | Very low |
| HU09 - Cerrar sesión | In Progress | auth | Very low |
| HU15 - Filtrar planes de simulación definidos por tipo | In Progress | simulation | Very low |
| HU38 - Adaptar simulador para ser multilingüe | Todo | usability | Very low |
| HU39 - Crear recursos para el español | Todo | usability | Very low |
| HU40 - Crear recursos para el inglés | Todo | usability | Very low |
| HU41 - Desplegar el simulador | Todo | | medium |

Figura 3.34: Pila del sprint 7.

### 3.7.2. Desarrollo del sprint

En primer lugar, se abordaron las historias de usuario pendientes del sprint 1 relacionadas con la autenticación del usuario, así como la historia de usuario destinada a filtrar los planes de simulación por tipo.

Respecto a las de autenticación del usuario, la interfaz visual para estas historias de usuario se había implementado en el primer sprint. En este momento, el objetivo consistía en integrar dicha interfaz con el servicio de autenticación interno de la empresa, permitiendo así la visualización de los detalles de la cuenta del usuario y la gestión de la sesión en el servidor.

La implementación se inspiró en la plataforma de control existente, buscando similitudes en la autenticación del usuario. Tras la correcta configuración de ciertos servicios y mecanismos de seguridad previamente no establecidos, se logró la funcionalidad deseada. En la figura 3.35 se muestra el resultado de la implementación de estas dos historias de usuario.

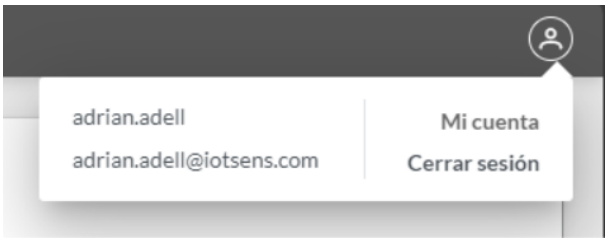



Figura 3.35: Menú de cuenta de usuario.

A continuación, en lo referente a la historia de filtrar planes por su tipo, aunque era operativa, cabe recordar que presentaba algunas inconsistencias en su uso, provocando problemas de usabilidad. Con la intención de solucionar dichos errores, se modificó el observador que le indicaba al menú del filtro cuando aparecer. Pese a que en el sprint 2 no fue posible solventar este problema, ahora, con toda la experiencia adquirida, fue relativamente sencilla.

Por otro lado, se abordaron las historias para conseguir un simulador plurilingüe. Para este desarrollo se utilizó una herramienta de Angular denominada Transloco[3], siendo necesario modificar todos los componentes visuales del simulador para que reflejaran el contenido según los archivos de configuración de idiomas. De este modo, la adición de nuevos idiomas requeriría únicamente la creación de un archivo de configuración correspondiente. Se implementó un servicio para determinar el idioma preferido del usuario, permitiendo la alternancia entre el español y el inglés, creándose los archivos de configuración necesarios para ambos idiomas. En la figura 3.36 se puede ver una de las pantallas en inglés.

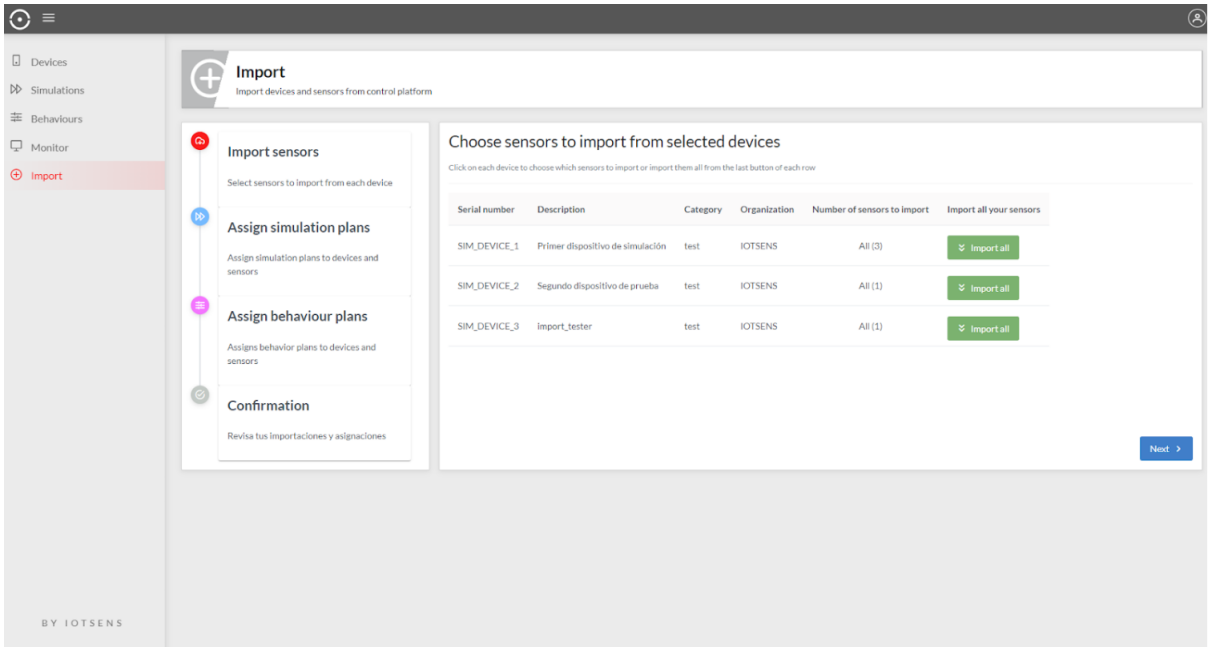



Figura 3.36: Pantalla de importar sensores en inglés.

Por último, quedaba pendiente desplegar el simulador en el entorno interno. Esta historia de usuario implicó realizar ajustes en el archivo *pom.xml* del proyecto del simulador y especificaciones adicionales en el repositorio de GitLab. Estas configuraciones permitieron el despliegue local del proyecto sin mayores inconvenientes, facilitando su uso por parte del equipo de desarrollo de software de la empresa.

Tras la finalización de las historias de usuario mencionadas y con tiempo disponible antes de concluir el sprint 7, consideré conveniente, con el fin de mejorar la interactividad y la experiencia del usuario, el rediseño de la sección *Monitor*.

Desde el principio, esta sección había sido utilizada para mostrar gráficamente todas las medidas simuladas. Al inicio era útil, sin embargo, en este punto del desarrollo carecía de sentido ya que mostrar todas las medidas del simulador en una gráfica no era práctico debido al gran volumen de datos generados en ese momento.

El objetivo del rediseño fue habilitar la visualización en tiempo real del estado global del simulador ya que la verificación individual de cada sensor resultaba ineficiente, y una visión global sería de gran utilidad tanto para los desarrolladores como para los usuarios finales.

La pantalla del *Monitor*, pasó de mostrar todas las medidas del simulador en una única gráfica, a visualizar la información relevante de lo que ocurría en un momento específico en el simulador. Para ello, la pantalla mostraba diferentes tarjetas, una para cada plan de simulación periódico, con información relativa a:

- Nombre del plan
- Frecuencia de muestreo del plan
- Sensores asignados
  - Detalles relevantes
  - El plan de comportamiento asignado
  - Última medida generada
- Tiempo estimado hasta la generación de nuevas medidas (cuenta atrás)

En la figura 3.37 se puede ver esta nueva sección *Monitor* rediseñada.

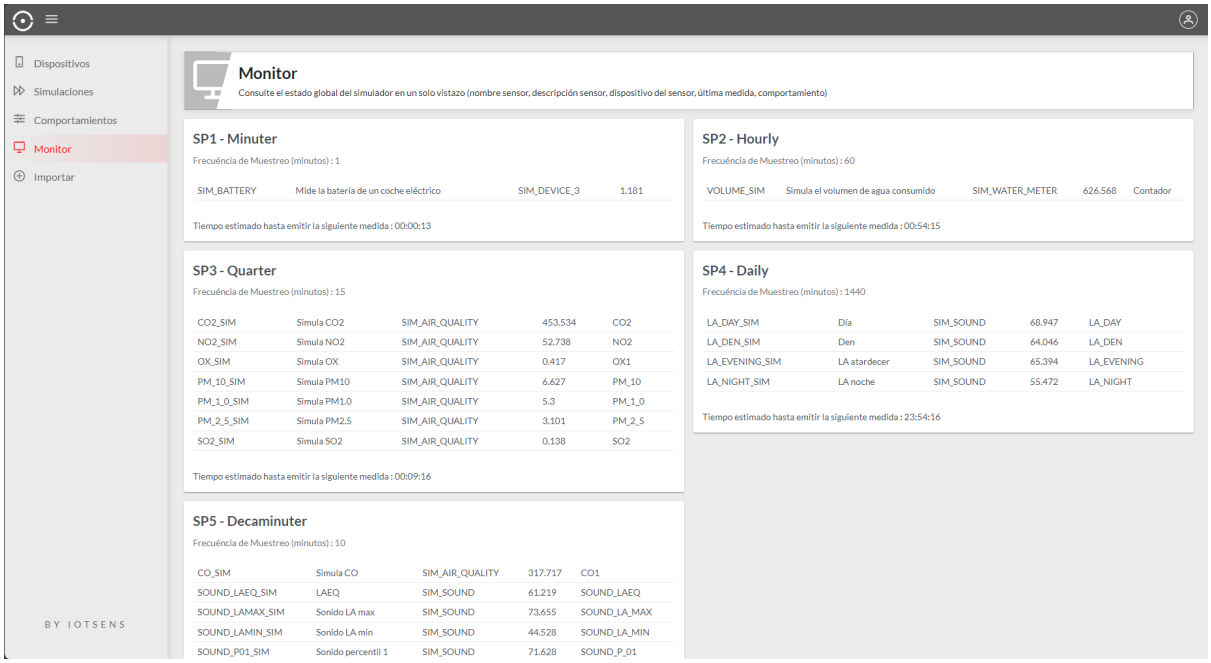



Figura 3.37: Nueva sección *Monitor*.

Para implementar esta nueva funcionalidad de la sección del *Monitor*, se reutilizaron endpoints previamente existentes, dado que la información requerida ya se utilizaba en otras secciones del simulador. El desafío consistió en actualizar en tiempo real el estado global del simulador, es decir, informar al frontend sobre las modificaciones en la base de datos para mostrar la última medida simulada de cada sensor y la cuenta regresiva hasta la generación de la siguiente medida.

Adicionalmente, identifiqué que los sensores podían seguir un tipo de comportamiento que no seguía ninguno de los patrones definidos hasta el momento. En respuesta a esta situación, decidí establecer un nuevo tipo de comportamiento denominado *latido*, que se sumaría a los tres ya existentes (aleatorio, ascendente y descendente). El término *latido* se escogió para este comportamiento debido a su similitud con el ritmo constante y periódico de los latidos del corazón o el *beat* de una canción, lo que facilita la comprensión de su funcionamiento y su aplicación práctica.

Para la implementación del nuevo tipo de comportamiento, se requirió la adición de una nueva opción en la creación o edición de un plan de comportamiento. Se tuvo que modificar el campo correspondiente al tipo de plan de comportamiento en la base de datos, con un nuevo script de Flyway, y se adaptó el generador de datos para soportar este nuevo tipo y generar medidas acordes a su definición.

Por otro lado, se reconoció que el proceso de edición de un plan de simulación, al ser idéntico al de creación, resultaba confuso. Este proceso permitía modificar el tipo de plan, ya sea periódico o de estrés, lo cual no era intuitivo para el usuario. Por consiguiente, se incorporó una historia de usuario adicional para simplificar la edición y evitar la posibilidad de cambiar el tipo de plan en el proceso de edición, puesto que tal opción carecía de sentido y solo generaba confusión.

Con la adición de estas tres nuevas historias de usuario, a continuación se presenta el estado de la pila del producto a mitad del séptimo sprint, así como la definición específica de las historias de usuario, en la figura 3.38.

| Title | Status | Scope | Priority |
|---|---|---|---|
| HU32 - Filtrar medidas simuladas de un sensor por plan de simulación | cancelled | generated data | low |
| HU42 - Perfeccionar proceso edición de los planes de simulación | Todo | simulation | low |
| HU43 - Añadir nuevo comportamiento a los planes de comportamiento | Todo | behaviour | high |
| HU44 - Rediseñar página del monitor | Todo | usability | Very low |

Figura 3.38: Estado de la pila del producto a mitad del sprint 7.

### 3.7.3. Detalles de la implementación

A continuación se detalla el desarrollo de las historias de usuario mencionadas:

- **HU08 y HU09 - Autenticación del Usuario:** con esta historia, además de acceder a los detalles de la cuenta, se permitía cambiar el idioma del simulador.
- **HU15 - Filtrado de Planes de Simulación:** aunque la implementación de esta historia de usuario era operativa, presentaba inconsistencias ya que a veces, al acceder a la sección de planes de simulación desde el menú lateral, se activaba directamente el filtro de planes, y en algunas ocasiones, era necesario realizar múltiples intentos para la correcta redirección, por lo que se aplicaron las correcciones necesarias para evitar este comportamiento.

- **HU38, HU39 y HU40 - Simulador Plurilingüe:** con estas implementaciones, añadir un nuevo idioma al simulador sería tan fácil como crear un nuevo archivo *JSON* (JavaScript Object Notation) de configuración para dicho idioma.
- **HU41 - Despliegue del Simulador:** gracias a estas configuraciones, cualquier actualización del simulador subida a GitLab se desplegará automáticamente en la versión más reciente.
- **HU42 - Optimización del Proceso de Edición:** la implementación de esta historia de usuario eliminó la selección del tipo de plan de simulación en el proceso de edición, manteniendo la opción de regresar a dicho paso. En caso de retroceder, solo se permite la selección del tipo de plan ya existente, acompañado de un mensaje explicativo.
- **HU43 - Nuevo Comportamiento de Sensores:** también se refactorizó esta parte de los generadores de medidas, abstrayendo en una nueva clase de ayuda los métodos comunes. Además, se crearon nuevas normas para que las medidas simuladas fueran más reales como, por ejemplo, no permitir demasiada variación entre las nuevas medidas sobre sus anteriores.
- **HU44 - Rediseño del Monitor:** aprovechando los conocimientos adquiridos para actualizar el simulador en tiempo real, también se implementó esta funcionalidad a tablas de medidas de sensores. De este modo, no hacía falta recargar la página para ver las nuevas medidas simuladas.

Con la implementación de estas historias de usuario, se concluyó el desarrollo del simulador, y la pila del producto quedó únicamente con la historia cancelada (HU32). En las Figuras 3.39 y 3.40, se muestran, respectivamente, la pila del producto final y el Kanban al término del desarrollo.

Los resultados finales tras las iteraciones sobre la pila del producto se detallarán en el capítulo 4.

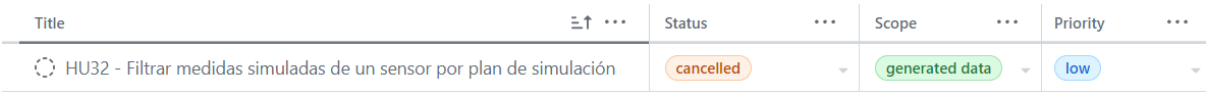



Figura 3.39: Pila del producto al final del sprint 7.

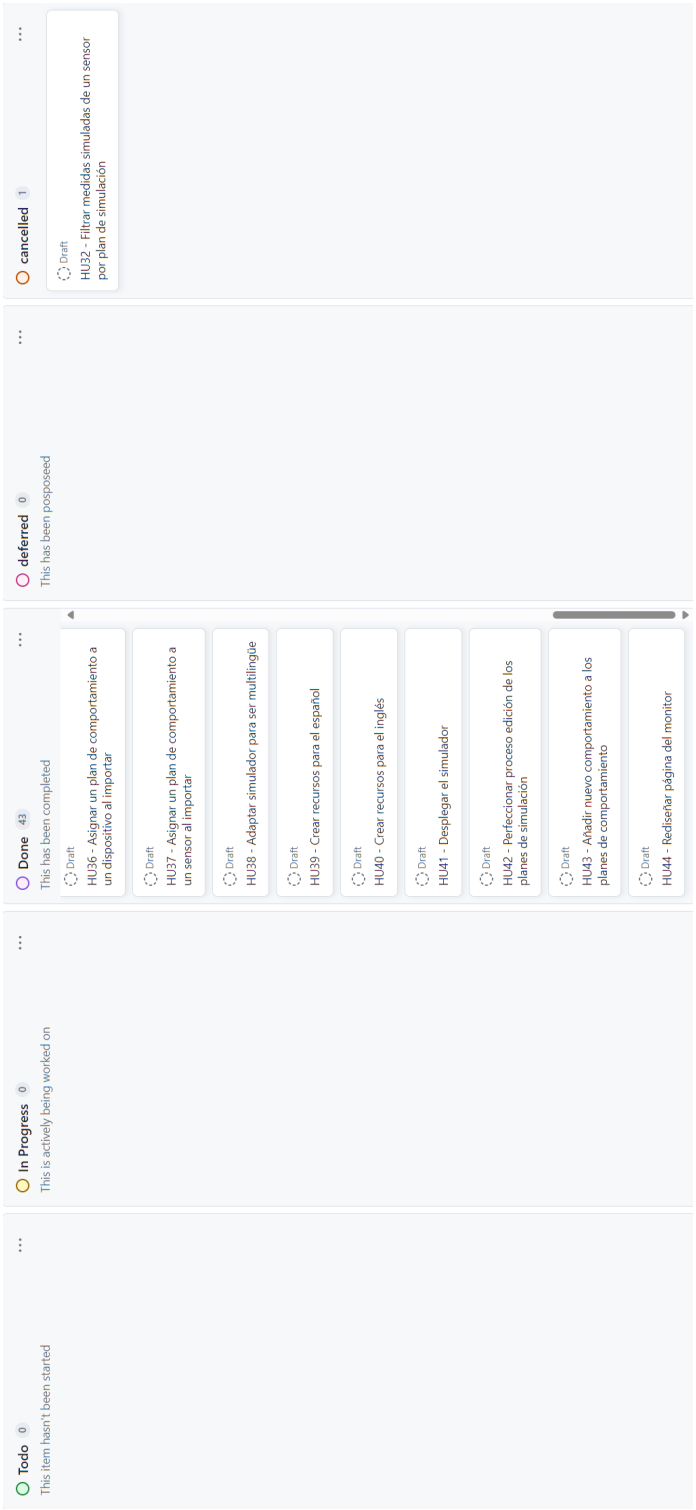



Figura 3.40: Estado del Kanban al final del sprint 7.

# Capítulo 4

# Resultados

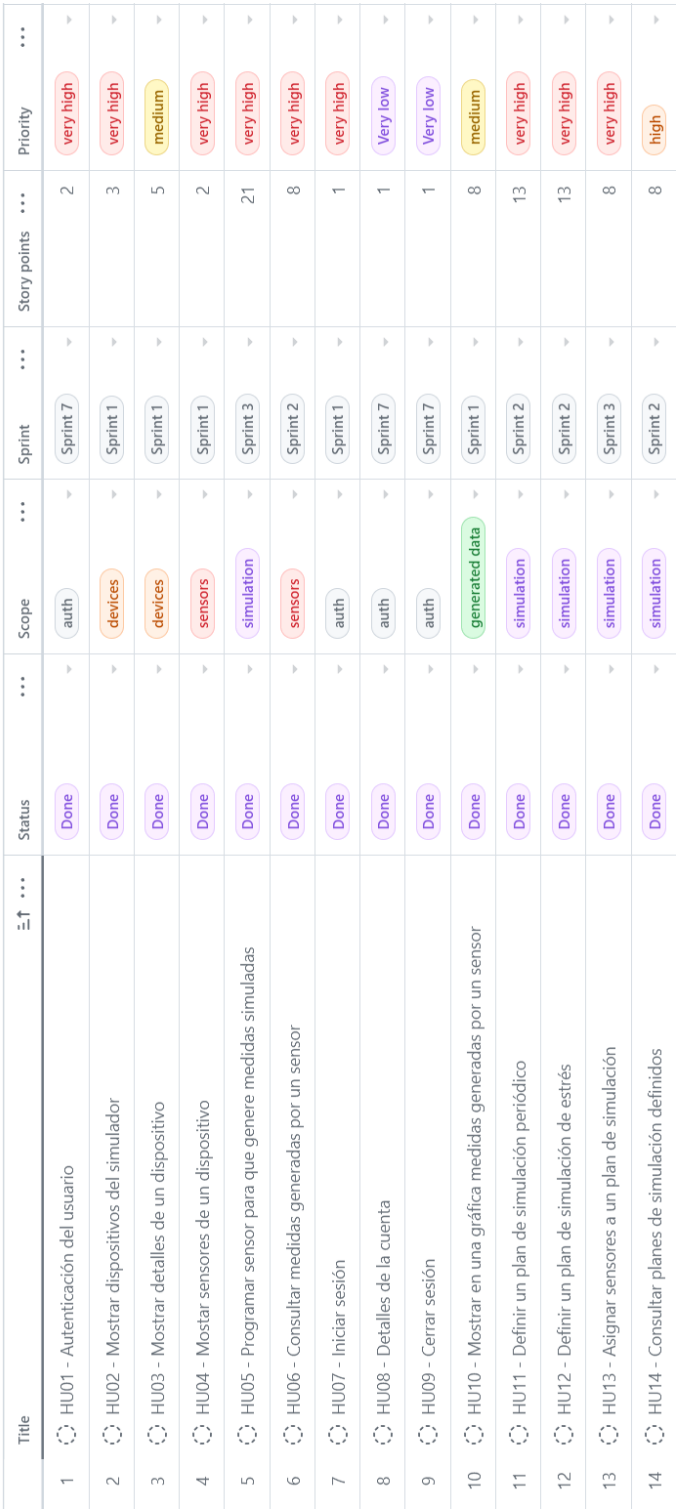
En este capítulo se detallan los resultados finales obtenidos tras todos los sprints descritos en el capítulo 3.

Con el fin de asegurar la calidad del producto obtenido, se han aplicado los principios descritos en libro Clean Code para obtener un código claro, comprensible y mantenible.

Además, durante el desarrollo, se han realizado demostraciones tras cada sprint al supervisor y otros miembros del equipo de trabajo para tener un control de calidad continuo sobre las funcionalidades incorporadas y comprobar que se ajustaban correctamente a lo requisitos, así como a la usabilidad del producto.

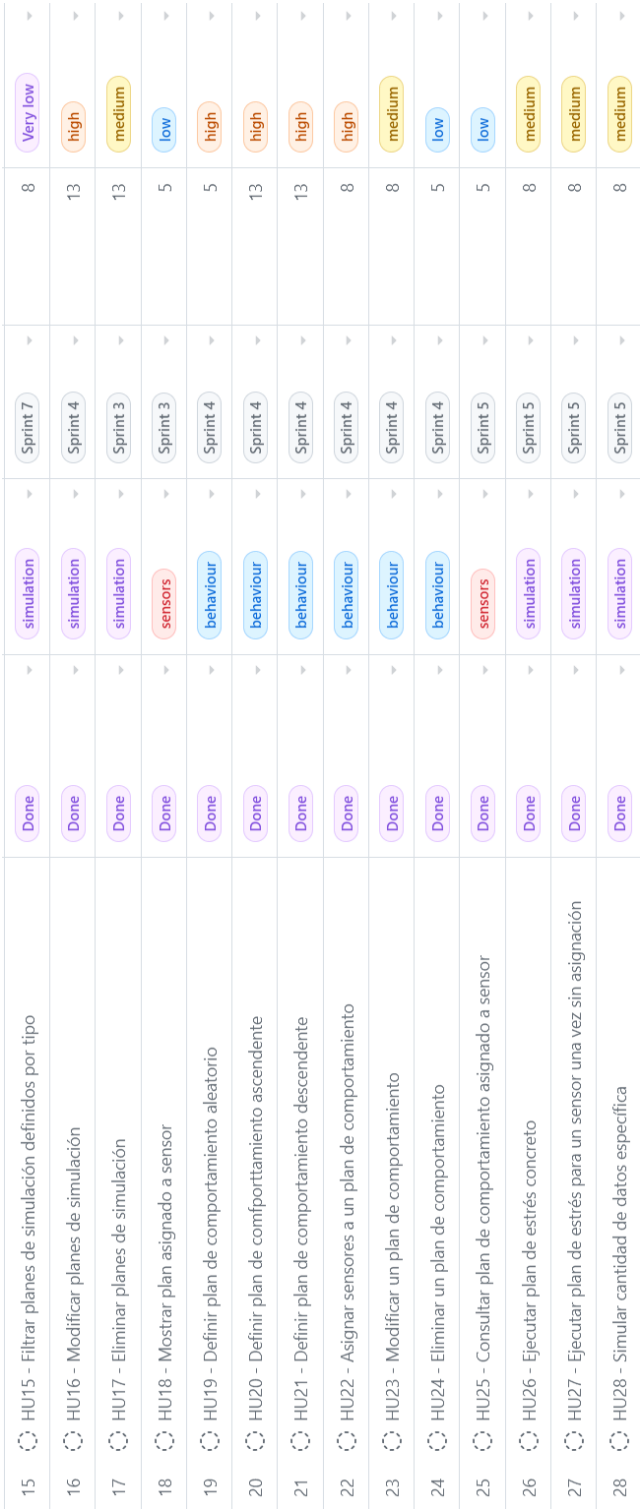
## 4.1. Pila del producto completa

Como se ha ido viendo a lo largo del documento, la pila del producto ha ido variando para adaptarse a las diferentes necesidades y requerimientos planteados por parte de la empresa. En la figuras que se muestran a continuación, 4.1, 4.2 y 4.3, se detalla la pila completa resultante de todas las iteraciones llevadas a cabo.



| | Title | Status | Scope | Sprint | Story points | Priority |
|---|---|---|---|---|---|---|
| 1 | HU01 - Autenticación del usuario | Done | auth | Sprint 7 | 2 | very high |
| 2 | HU02 - Mostrar dispositivos del simulador | Done | devices | Sprint 1 | 3 | very high |
| 3 | HU03 - Mostrar detalles de un dispositivo | Done | devices | Sprint 1 | 5 | medium |
| 4 | HU04 - Mostar sensores de un dispositivo | Done | sensors | Sprint 1 | 2 | very high |
| 5 | HU05 - Programar sensor para que genere medidas simuladas | Done | simulation | Sprint 3 | 21 | very high |
| 6 | HU06 - Consultar medidas generadas por un sensor | Done | sensors | Sprint 2 | 8 | very high |
| 7 | HU07 - Iniciar sesión | Done | auth | Sprint 1 | 1 | very high |
| 8 | HU08 - Detalles de la cuenta | Done | auth | Sprint 7 | 1 | Very low |
| 9 | HU09 - Cerrar sesión | Done | auth | Sprint 7 | 1 | Very low |
| 10 | HU10 - Mostrar en una gráfica medidas generadas por un sensor | Done | generated data | Sprint 1 | 8 | medium |
| 11 | HU11 - Definir un plan de simulación periódico | Done | simulation | Sprint 2 | 13 | very high |
| 12 | HU12 - Definir un plan de simulación de estrés | Done | simulation | Sprint 2 | 13 | very high |
| 13 | HU13 - Asignar sensores a un plan de simulación | Done | simulation | Sprint 3 | 8 | very high |
| 14 | HU14 - Consultar planes de simulación definidos | Done | simulation | Sprint 2 | 8 | high |

Figura 4.1: Primera parte de la pila del producto completa.



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 15 | HU15 - Filtrar planes de simulación definidos por tipo | Done | simulation | Sprint 7 | 8 | Very low |
| 16 | HU16 - Modificar planes de simulación | Done | simulation | Sprint 4 | 13 | high |
| 17 | HU17 - Eliminar planes de simulación | Done | simulation | Sprint 3 | 13 | medium |
| 18 | HU18 - Mostrar plan asignado a sensor | Done | sensors | Sprint 3 | 5 | low |
| 19 | HU19 - Definir plan de comportamiento aleatorio | Done | behaviour | Sprint 4 | 5 | high |
| 20 | HU20 - Definir plan de comportamiento ascendente | Done | behaviour | Sprint 4 | 13 | high |
| 21 | HU21 - Definir plan de comportamiento descendente | Done | behaviour | Sprint 4 | 13 | high |
| 22 | HU22 - Asignar sensores a un plan de comportamiento | Done | behaviour | Sprint 4 | 8 | high |
| 23 | HU23 - Modificar un plan de comportamiento | Done | behaviour | Sprint 4 | 8 | medium |
| 24 | HU24 - Eliminar un plan de comportamiento | Done | behaviour | Sprint 4 | 5 | low |
| 25 | HU25 - Consultar plan de comportamiento asignado a sensor | Done | sensors | Sprint 5 | 5 | low |
| 26 | HU26 - Ejecutar plan de estrés concreto | Done | simulation | Sprint 5 | 8 | medium |
| 27 | HU27 - Ejecutar plan de estrés para un sensor una vez sin asignación | Done | simulation | Sprint 5 | 8 | medium |
| 28 | HU28 - Simular cantidad de datos específica | Done | simulation | Sprint 5 | 8 | medium |

Figura 4.2: Segunda parte de la pila del producto completa.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 29 | HU29 - Guardar medidas simuladas en base de datos central | Done | generated data | Sprint 5 | 21 | very high |
| 30 | HU30 - Importar dispositivos de base de datos central | Done | devices | Sprint 5 | 21 | very high |
| 31 | HU31 - Importar sensores de un dispositivo | Done | sensors | Sprint 6 | 8 | high |
| 32 | HU32 - Filtrar medidas simuladas de un sensor por plan de simulación | cancelled | generated data | | 13 | low |
| 33 | HU33 - Asignar planes a las importaciones | Done | simulation | Sprint 6 | 21 | high |
| 34 | HU34 - Asignar plan de simulación a un dispositivo al importar | Done | simulation | Sprint 6 | 8 | high |
| 35 | HU35 - Asignar un plan de simulación a un sensor al importar | Done | simulation | Sprint 6 | 8 | high |
| 36 | HU36 - Asignar un plan de comportamiento a un dispositivo al importar | Done | simulation | Sprint 6 | 8 | high |
| 37 | HU37 - Asignar un plan de comportamiento a un sensor al importar | Done | simulation | Sprint 6 | 8 | high |
| 38 | HU38 - Adaptar simulador para ser multilingüe | Done | usability | Sprint 7 | 3 | Very low |
| 39 | HU39 - Crear recursos para el español | Done | usability | Sprint 7 | 3 | Very low |
| 40 | HU40 - Crear recursos para el inglés | Done | usability | Sprint 7 | 3 | Very low |
| 41 | HU41 - Desplegar el simulador | Done | | Sprint 7 | 8 | medium |
| 42 | HU42 - Perfeccionar proceso edición de los planes de simulación | Done | simulation | Sprint 7 | 3 | low |
| 43 | HU43 - Añadir nuevo comportamiento a los planes de comportamiento | Done | behaviour | Sprint 7 | 8 | high |
| 44 | HU44 - Rediseñar página del monitor | Done | usability | Sprint 7 | 13 | Very low |

Figura 4.3: Tercera parte de la pila del producto completa.

## 4.2. Diseño de la base de datos

El diseño de la base de datos es un componente crucial de cualquier proyecto de desarrollo de software. En este proyecto, el diseño de la base de datos ha pasado por varias iteraciones y cambios a lo largo de los sprints. Cada cambio ha sido una respuesta a los requisitos cambiantes y a las necesidades emergentes del proyecto.

En esta sección, se detallan los resultados de este proceso de diseño mostrando su estado final, que representa la culminación de todo el trabajo de diseño realizado durante el proyecto.

### 4.2.1. Diseño Conceptual

En la figura 4.4 se puede observar el diseño conceptual de la base de datos del simulador. Este consiste en una representación abstracta de la base de datos, donde se definen las entidades, sus atributos, así como las relaciones entre ellas. Este nivel se enfoca en la estructura de alto nivel de como se organizarán los datos, sin entrar en detalles técnicos de almacenamiento.

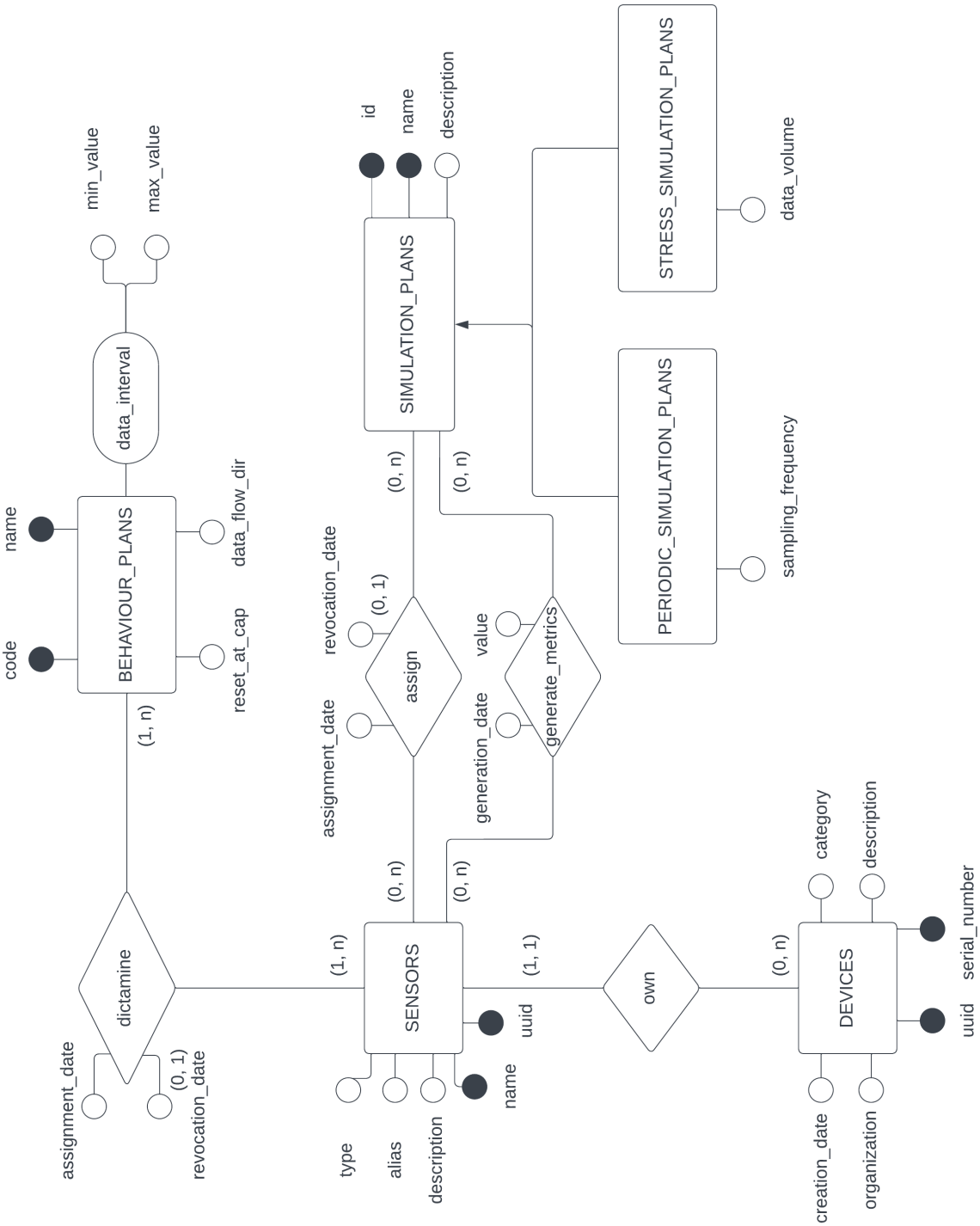



Figura 4.4: Diseño Conceptual de la Base de Datos.

### 4.2.2. Diseño Lógico

El diseño lógico, que se puede ver en la figura 4.5, detalla cómo se implementará el diseño conceptual en el sistema de gestión de bases de datos específico. Incluye la definición de tablas, columnas, tipos de datos y restricciones, y es un paso crucial para la correcta creación del diseño físico de la base de datos.

**devices**(uuid, serial_number, description, category, organization, creation_date)

serial_number es clave alternativa

**sensors**(uuid, name, description, alias, type, device_uuid)

name se clave alternativa

sensors —device_uuid—> devices No | Restrict | Cascade

**simulation_plans**(id, name, description)

name es clave alternativa

**periodic_simulation_plans**(id, sampliong_frecuency)

periodic_simulation_plans —id—> simulation_plans No | Restrict | Cascade

**stress_simulation_plans**(id, data_volume)

stress_simulation_plans —id—> simulation_plans No | Restrict | Cascade

**assign**(sensor_uuid, plan_id, assingment_date, revocation_date)

revocation_date acepta nulos

assign —sensor_uuid—> sensors No | Restrict | Cascade

assign —plan_id—> simulation_plans No | Restrict | Cascade

**generate_metrics**(sensor_uuid, plan_id, generation_date, value)

generate_metrics—sensor_uuid—> sensors No | Restrict | Cascade

generate_metrics—plan_id—> simulation_plans No | Restrict | Cascade

**behaviour_plans**(code, name, min_value, max_value, data_flow_dir, reset_at_cap)

name es clave alternativa

**dictamine**(behaviour_code, sensor_uuid, assingment_date, revocatioin_date)

revocation_date acepta nulos

dictamine —behvaiour_code—> behaviour_plans No | Restrict | Cascade

dictamine —sensor_uuid—> sensors No | Restrict | Cascade

Figura 4.5: Diseño Lógico de la Base de Datos del Simulador.

## 4.3. Diagramas de actividades

Con el fin de facilitar la comprensión del funcionamiento del simulador, a continuación se incluyen varios diagramas de actividades que ilustran los flujos principales del simulador. Cabe destacar que para interactuar con el simulador hay que tener iniciada la sesión con una cuenta de IoTsens.

Sin embargo, antes de presentar los diagramas de actividades, en la figura 4.6 se puede observar un árbol con las secciones principales que tiene el simulador, todas ellas accesibles desde en el menú lateral de la aplicación. Los diagramas de actividades empiezan a partir de cada una de estas secciones, evitando así la necesidad de repetir para cada uno el proceso de login y redirección a la sección conveniente usando el *sidebar*. Por último, mencionar que la sección de dispositivos es la que se muestra por defecto al acceder al simulador.


Figura 4.6: Secciones del sidebar del simulador.

Una vez detallado el árbol de navegación del simulador, en las figuras 4.7, 4.8 y 4.9 se muestran los diagramas de actividades.

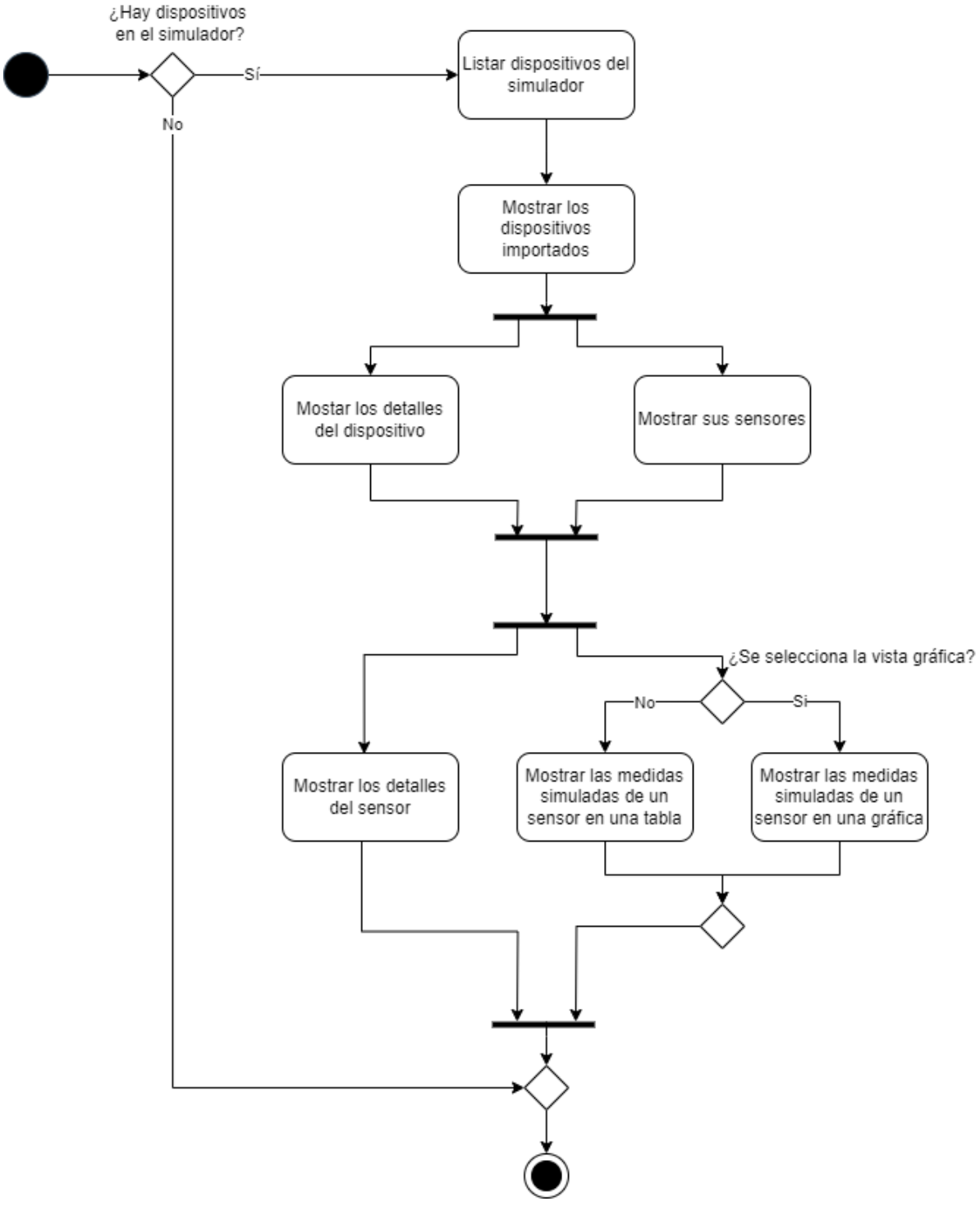



Figura 4.7: Diagrama de Actividades del flujo de Dispositivos.

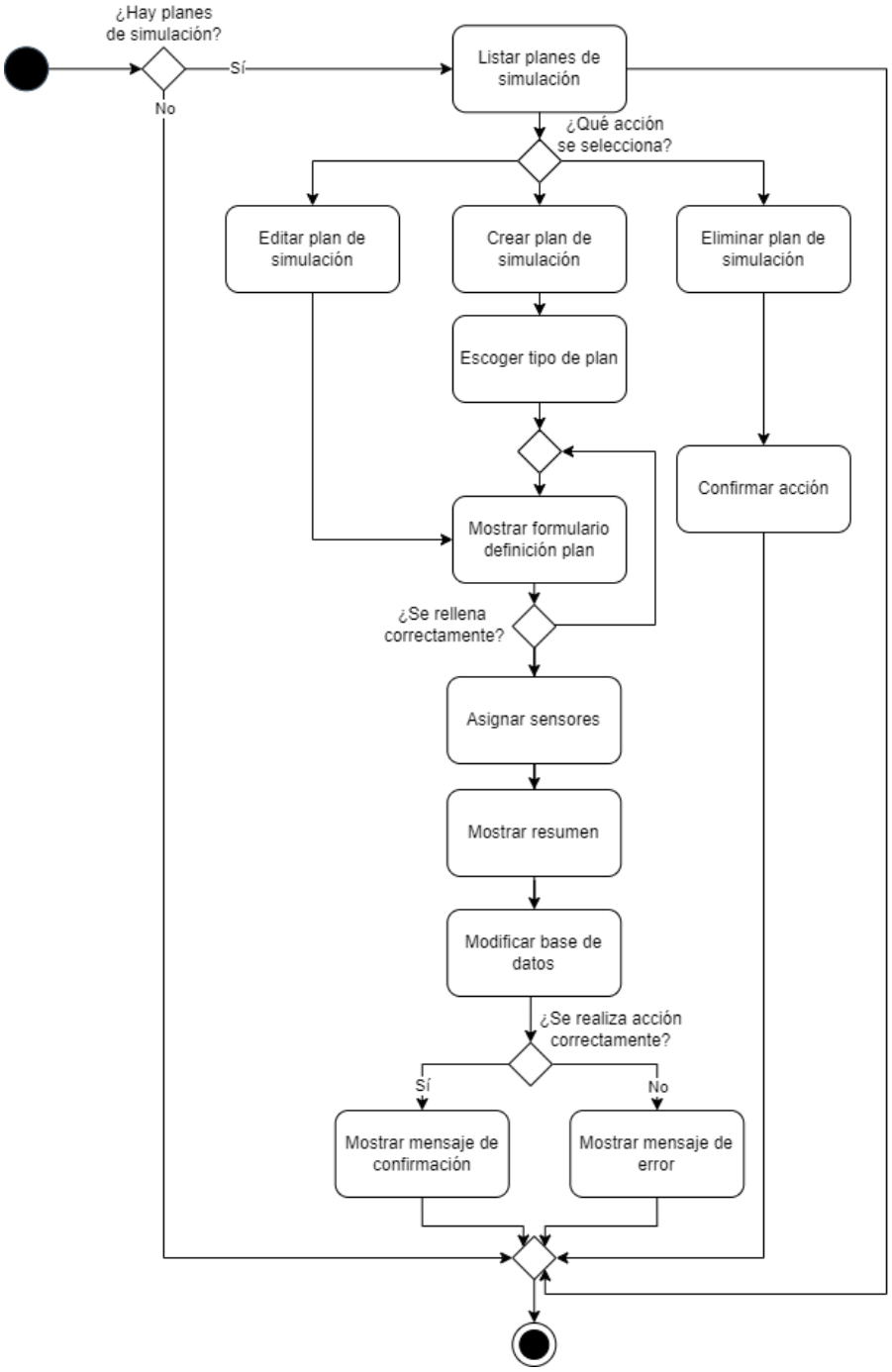



Figura 4.8: Diagrama de Actividades del flujo de Simulación.

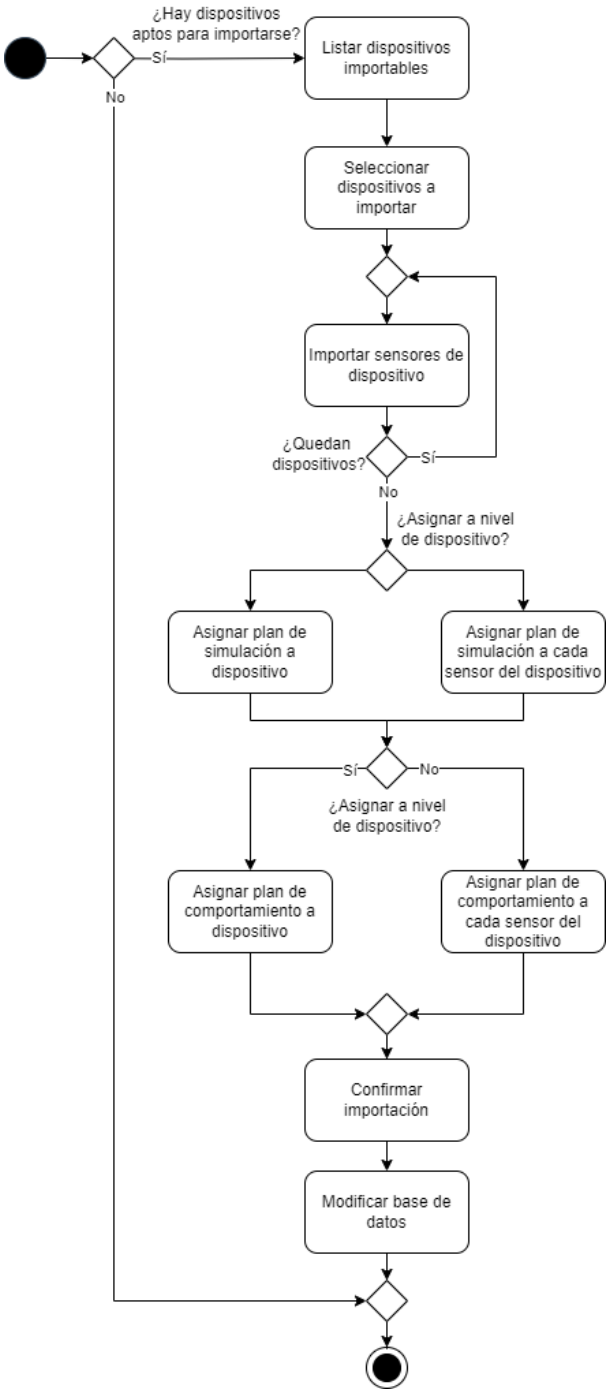



Figura 4.9: Diagrama de Actividades del flujo de Importar.

## 4.4. Tests de pruebas

### 4.4.1. Plantilla tests de aceptación

Para asegurar el correcto funcionamiento del simulador, se ha creado una plantilla de tests de aceptación, como se puede ver en la figura 4.10, sobre varios segmentos funcionales de la aplicación. Estas pruebas deben ser ejecutadas manualmente por un QA (Quality Assurance) que proporcionará el feedback oportuno sobre funcionalidad a evaluar.

| Id | HU Requisito | Propósito | Pre Condición | Test Pasos | Test Data | Resultado Esperado |
|---|---|---|---|---|---|---|
| HU18-E1 | HU018 | Probar que el proceso de creación de planes de simulación periódicos funciona | Crear un plan de simulación periódico inexistente | 1 Iniciar sesión<br>2 Seleccionar Simulaciones en el sidemenu<br>3 Crear plan<br>4 Escoger el tipo periódico<br>5 Establecer detalles del plan<br>6 Asignar sensores<br>7 Confirmar<br>8 Verificar el plan se ha añadido a la lista | Nombre: TEST_PERIODIC_PLAN_1<br>Descripcion: Plan periódico 1<br>Frecuencia de muestreo: 1<br>Asginar sensor: TEST_SENSOR | El plan se crea correctamente y se muestra en la lista de planes |
| HU18-E2 | HU018 | Probar que el proceso de creación de planes de simulación periódicos no permite continuar con detalles inválidos sobre el plan de simulación | Crear un plan de simulación periódico con detalles inválidos | 1 Iniciar sesión<br>2 Seleccionar Simulaciones en el sidemenu<br>3 Crear plan<br>4 Escoger el tipo periódico<br>5 Intentar continuar sin rellenar ningún input<br>6 Establecer detalles inválidos del plan<br>7 Inentar ir al siguiente paso<br>8 Corregir errores detectados por la UI<br>9 Continuar con la creación | Nombre invalido: TE<br>Descripcion: aa<br>Frecuencia de muestreo inválida: -1<br><br>Los datos válidos son los mismos que en el test HU18-E1 | No permite continuar hasta que se insertan datos válidos (no se pretende crear el plan) |

Figura 4.10: Captura parcial de la plantilla de tests de aceptación generada en MS Excel.

### 4.4.2. Tests sobre generadores

Los generadores constituyen un componente esencial del código, ya que son responsables de producir los datos para los sensores. Por tanto, es crucial verificar su correcto funcionamiento. Para ello se implementó una batería de tests de integración acotados.

Cabe mencionar, que estos tests solo son aplicables a los planes de simulación de estrés, puesto que son los que permiten generar una cantidad considerable de datos en tiempo 0. Aún así, para los planes de simulación periódicos, se usa el mismo algoritmo generador, no precisando de pruebas específicas ya que se asume su correcto funcionamiento y la validez de las medidas generadas al tener los otros planes comprobados.

En el fragmento de código 4.1 se puede ver un ejemplo de un test. En este caso, para probar que la secuencia creada por el generador es ascendente.

Código 4.1: Test para generar una secuencia ascendente.

```
1  @Test
2  public void testGenerateAscendingSequence() {
3      List<Float> sequence = stressValuesGenerator.
           ↪ generateSequenceOfNumbers(testMinValue, testMaxValue,
           ↪  true, DataFlowDir.ASC, 1000, null);
4      assertNotNull(sequence);
5      assertEquals(1000, sequence.size());
6      for(int i = 0; i < sequence.size() - 1; i++) {
7          if (sequence.get(i) > sequence.get(i + 1)) {
8              assertTrue(isResetValue(sequence.get(i + 1)));
9          } else {
10             assertTrue(sequence.get(i) <= sequence.get(i + 1))
                   ↪ ;
11         }
12     }
13 }
```

# Capítulo 5

# Conclusiones

## 5.1. Objetivos alcanzados

En un primer momento, no estaba claro como se quería simular un sensor. Las expectativas iniciales planteadas por el supervisor de la empresa eran generar medidas aleatorias cada cierto tiempo y conectar con su base de datos y dispositivos. Sin embargo, a medida que avanzaba el proyecto, se fue perfilando y ampliando el análisis y el diseño del simulador hasta llegar a un producto final mucho más completo de lo que se esperaba en un primer momento.

A pesar de la incertidumbre inicial sobre el alcance del producto, se ha conseguido una solución que satisface todas las funcionalidades necesarias y se han superado con éxito los diferentes desafíos que surgieron durante el desarrollo.

El resultado es un simulador eficiente que cumple ampliamente con los requisitos especificados, capaz no solo de establecer la frecuencia y generar un gran volumen de datos, sino que también permite definir su comportamiento en base a los parámetros deseados.

Lo que comenzó como una simple funcionalidad adicional para su aplicación central, ha evolucionado hasta convertirse en una aplicación independiente pero integrada, que interactúa de manera efectiva con el ecosistema IoTsens.

En resumen, se puede concluir que se han superado los objetivos especificados.

## 5.2. Conocimientos Aplicados

En el desarrollo de este proyecto, se han aplicado diversos conocimientos y técnicas de programación. A continuación, se detallan algunos de ellos.

### 5.2.1. Análisis de requisitos

El análisis de requisitos es una fase crucial en el desarrollo de cualquier proyecto de software. En esta fase, se identifican y documentan las necesidades y expectativas de la empresa para definir claramente lo que el sistema debe hacer. Sin embargo, en este caso, como ya se ha mencionado a lo largo de la memoria, este análisis ha ido completándose en diferentes etapas a lo largo del desarrollo del proyecto e identificándose de forma paulatina dichas necesidades y expectativas.

En el caso de este proyecto, se ha aplicado el concepto de Historias de Usuario descritas en capítulos anteriores, que han permitido definir los requisitos del simulador de una manera que se centra en el valor entregado al usuario, facilitando la comprensión de lo que se necesita construir y por qué. Todas ellas se han estimado, puntuado y priorizado.

Además, las Historias de Usuario fomentan la comunicación entre los desarrolladores y los *stakeholders*, ayudando a asegurar que todos tengan una comprensión compartida de los objetivos del proyecto, vital en este proyecto tan poco predefinido.

Todos estos conocimientos, técnicas y recomendaciones, se han estudiado en diferentes asignaturas cursadas durante el grado.

### 5.2.2. Diseño e implementación de bases de datos

Durante la formación académica, se han visto en profundidad los conceptos de diseño e implementación de bases de datos.

Se ha utilizado SQL, un lenguaje de programación de bases de datos que también se ha estudiado durante el grado. SQL permite definir, manipular y consultar datos en una base de datos relacional.

El diseño de la base de datos se ha realizado siguiendo las prácticas aprendidas, incluyendo la normalización para minimizar la redundancia de datos y mejorar la integridad de los mismos. Se han definido tablas y relaciones para almacenar eficientemente los datos necesarios para el simulador.

En resumen, los conceptos de diseño e implementación de bases de datos aprendidos durante los diferentes cursos académicos han sido fundamentales para el desarrollo de este proyecto.

### 5.2.3. Patrones de Diseño

Se han utilizado varios patrones de diseño, vistos conceptualmente en asignaturas del grado, para estructurar el código de manera eficiente y mantenible.

#### Singleton

El patrón Singleton se ha utilizado para garantizar que una clase tenga una única instancia, proporcionando un punto de acceso global a ella. Este patrón ha sido especialmente útil para coordinar acciones a través del sistema en procesos multietapa, como la creación y edición de planes de simulación y comportamiento.

Estos procesos multietapa requieren un almacenamiento común entre cada uno de esos pasos. Al ser los servicios Singleton, se garantiza que todos los pasos de un proceso acceden a la misma instancia de ese proceso, lo que permite un almacenamiento común de los datos necesarios, asegurando la coherencia y evitando conflictos, permitiendo que los datos se mantengan y se utilicen de manera eficiente a lo largo de las diferentes etapas del proceso.

#### Observador

El patrón Observador se ha utilizado para permitir que los objetos se suscriban a eventos y sean notificados cuando estos ocurren.

Este patrón ha sido fundamental para manejar interacciones complejas entre componentes. Por ejemplo, se utilizó para notificar al frontend cuando se habían almacenado nuevas medidas simuladas en la base de datos. Esto permitió actualizar las tablas de medidas de los sensores y la página del monitor en tiempo real, mejorando la experiencia del usuario.

### 5.2.4. Clean Code

Se ha seguido la filosofía de Clean Code y otras buenas prácticas, asimiladas a lo largo de los cursos universitarios, para mantener el código limpio y legible. Esto incluye principios como:

- Elegir nombres descriptivos y significativos para variables y funciones.
- Mantener las funciones y clases pequeñas y enfocadas.
- Evitar la duplicación de código mediante la reutilización y la abstracción.
- Usar comentarios solo cuando sea necesario para explicar el *por qué*, no el *qué*.

Además, se ha llevado a cabo una generalización de las clases y componentes con el objetivo de maximizar su reutilización. Esto significa que se han diseñado y estructurado de tal manera

que pueden ser empleadas en múltiples contextos y para diversas funcionalidades, sin la necesidad de crear nuevas instancias o duplicar código. Esta práctica no solo mejora la eficiencia del desarrollo y reduce la posibilidad de errores, sino que también contribuye a un código más limpio y mantenible. En resumen, la reutilización de clases y componentes ha sido una estrategia clave en el desarrollo de este proyecto.

Estos conocimientos y técnicas han sido fundamentales para el éxito del proyecto.

## 5.3. Mejoras futuras

Se ha conseguido implementar un simulador bastante completo con una base sólida que permite futuras ampliaciones con un esfuerzo mínimo. De hecho, es en esta área de expansión donde reside el potencial para mejoras futuras, particularmente en la ampliación de las capacidades de personalización de los parámetros que definen los comportamientos de las simulaciones.

## 5.4. Reflexiones finales

Este proyecto ha sido una gran oportunidad para aplicar y consolidar los conocimientos adquiridos durante el grado. La puesta en práctica de estos conocimientos ha permitido no solo afianzarlos, sino también comprenderlos a un nivel más profundo.

Además, la gran autonomía otorgada durante el proyecto ha sido una experiencia de aprendizaje en sí misma. Me ha permitido aprender a tomar decisiones informadas y efectivas, a menudo en situaciones de incertidumbre.

También se ha fomentado las habilidades de trabajo en equipo, ya que se ha tenido que colaborar estrechamente con otros miembros del departamento para alcanzar los objetivos del proyecto.

Finalmente, me gustaría expresar mi agradecimiento al supervisor de las prácticas. Su orientación y apoyo han sido de gran ayuda durante todo el proyecto, demostrando en todo momento la confianza depositada en mí, lo que ha conseguido que esté motivado e implicado en el proyecto. También, agradecer a todo el equipo que me ha ayudado a alcanzar este resultado final. Sin olvidar que me han facilitado la integración en el entorno laboral, haciéndome sentir uno más del equipo.

En resumen, este proyecto ha sido una experiencia que me ha permitido aplicar y profundizar los conocimientos adquiridos durante la carrera, al tiempo que he desarrollado habilidades valiosas para el futuro.

# Bibliografía

[1] Aditya Chaturvedi. Convergence of technologies undergird the smart city revolution. https://www.geospatialworld.net/blogs/technologies-undergird-smart-city-revolution/. [Consulta: 7 de Marzo de 2024].

[2] Alexander Shvets. Patrones de diseño de software: Observador. https://refactoring.guru/es/design-patterns/observer. [Consulta: 29 de Febrero de 2024].

[3] Anartz Mugika Ledo. Internacionalización en un proyecto angular — transloco [2/4]. https://mugan86.medium.com/internacionalizaci%C3%B3n-en-un-proyecto-angular-transloco-2-4-2e55d66ce1e2. [Consulta: 6 de Marzo de 2024].

[4] Andrew Djandrw. ¿qué es scrum? https://medium.com/@andrewdjandrw/qu%C3%A9-es-scrum-674c6b791af4. [Consulta: 7 de Marzo de 2024].

[5] Asana. ¿qué es la metodología kanban y cómo funciona? https://asana.com/es/resources/what-is-kanban. [Consulta: 17 de Marzo de 2024].

[6] freeCodeCamp. Javascript debounce example. https://www.freecodecamp.org/news/javascript-debounce-example/. [Consulta: 15 de Marzo de 2024].

[7] Grupo Gimeno. Empresas del grupo gimeno. https://www.grupogimeno.com/grupo-gimeno/. [Consulta: 7 de Marzo de 2024].

[8] Highsoft. Highcharts official page. https://www.highcharts.com/demo. [Consulta: 12 de Marzo de 2024].

[9] Idital. Diccionario seo: Lazy loading. https://idital.com/diccionario-seo/lazy-loading/. [Consulta: 24 de Febrero de 2024].

[10] IoTSens. Iotsens. https://www.iotsens.com/. [Consulta: 10 de Febrero de 2024].

[11] IoTsens. Roadmap de iotsens. https://www.iotsens.com/compania/. [Consulta: 7 de Marzo de 2024].

[12] Dantes Lahens. What is spike in scrum? a reliable agile technique. https://danteslahens.com/what-is-spike-in-scrum/. [Consulta: 17 de Marzo de 2024].

[13] Mailchimp. ¿qué es un endpoint (punto final) api? https://mailchimp.com/es/resources/what-is-an-api-endpoint/. [Consulta: 6 de Marzo de 2024].

[14] Paraschiv, Eugen. Spring scheduled tasks. https://www.baeldung.com/spring-scheduled-tasks. [Consulta: 14 de Febrero de 2024].

[15] Repsol S.A. Smart cities: ciudades inteligentes para un futuro sostenible. https://www.repsol.com/es/energia-futuro/tecnologia-innovacion/smart-cities/index.cshtml. [Consulta: 10 de Febrero de 2024].

[16] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. [Consulta: 6 de Marzo de 2024].

[17] Sutherland, Jeff and Schwaber, Ken. The scrum guide. https://scrumguides.org/. [Consulta: 8 de Febrero de 2024].

[18] Wikipedia. Sucesión de fibonacci. https://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci. [Consulta: 17 de Marzo de 2024].