

# **Una solución para la industria 4.0**

**Antonio José Puerto Borreguero**

**Trabajo fin de Grado**

**Supervisado por Dr. Ángel Jesús Varela Vaca and  
Dr. Álvaro Valencia Parra**



**Universidad de Sevilla**

**noviembre 2022**

Publicado en noviembre 2022 por  
Antonio José Puerto Borreguero  
Copyright © MMXXII  
[antoniopuerto8@gmail.com](mailto:antoniopuerto8@gmail.com)

Realizado en Noviembre de 2022

Yo, D. Antonio José Puerto Borreguero con NIF número 29510356E,

**DECLARO**

mi autoría del trabajo que se presenta en la memoria de este trabajo fin de grado que tiene por título:

*Una solución para la industria 4.0*

Lo cual firmo,

Fdo. D. Antonio José Puerto Borreguero  
en la Universidad de Sevilla  
28/11/2022

## Resumen

Este proyecto pretende mostrar una posible solución a una empresa cuyo objetivo es digitalizar la misma. Se abordarán tecnologías que nos permitan recolectar datos a través de dispositivos y analizarlos para poder tomar decisiones. En nuestro caso, estudiaremos si es rentable que un producto siga en el proceso de producción cuando no ha cumplido con los requisitos de calidad que el modelo de aprendizaje automático ha establecido. Para esto se utilizarán modelos de inteligencia artificial que nos permitan determinar en cada fase del proceso si el producto cumple con los requisitos de calidad, esto nos permite tomar decisiones en tiempo real sobre qué acción realizar en cada momento. Por ejemplo, no tiene sentido seguir invirtiendo recursos materiales y económicos en un producto que no se podrá vender por qué será desecharlo. El objetivo principal no es ofrecer una solución a una empresa para generar más ingresos, sino optimizar el sistema ya existente para anticiparnos a los errores y no gastar recursos innecesarios. Por último, se proporcionará también una aplicación que nos permita visualizar de forma cómoda los datos en tiempo real de la empresa y todos sus procesos.

# Índice general

<b>I</b>	<b>Introducción</b>	<b>3</b>
	<b>1. Contexto</b>	<b>5</b>
	1.1. Introducción . . . . .	6
	1.2. Estado del arte . . . . .	8
	1.3. Riesgos del proyecto . . . . .	9
	<b>2. Objetivos</b>	<b>11</b>
	2.1. Motivación . . . . .	12
	2.2. Lista de objetivos . . . . .	12
<b>II</b>	<b>Análisis y Diseño</b>	<b>15</b>
	<b>3. Análisis de requisitos</b>	<b>17</b>
	3.1. Requisitos de información . . . . .	18
	3.2. Requisitos funcionales . . . . .	22
	3.3. Requisitos no funcionales . . . . .	23
	<b>4. Diseño del sistema</b>	<b>25</b>
	4.1. Casos de uso . . . . .	26
	4.2. Procesos . . . . .	29
	4.2.1. Etapa de Proveedores . . . . .	29
	4.2.2. Etapa de transporte a la fabrica . . . . .	29

4.2.3. Etapa de fabricación . . . . .	30
4.2.4. Etapa de transporte al almacén . . . . .	30
4.2.5. Etapa de almacenamiento . . . . .	31
4.3. Modelo de datos . . . . .	32
4.4. Arquitectura lógica del sistema . . . . .	33
<b>III Desarrollo del proyecto</b>	<b>35</b>
<b>5. Desarrollo e implementacion</b>	<b>37</b>
5.1. Arquitectura del sistema . . . . .	38
5.2. Tecnologías y herramientas utilizadas . . . . .	39
5.3. Dispositivo IoT . . . . .	40
5.4. Generación y procesamiento de los datos . . . . .	50
5.4.1. Generación de datos . . . . .	50
5.4.2. Procesamiento de datos . . . . .	60
5.5. Back-end . . . . .	68
5.6. Aplicación multiplataforma . . . . .	73
<b>IV Pruebas</b>	<b>77</b>
<b>6. Pruebas</b>	<b>79</b>
6.1. Prueba 1: Dispositivo IoT . . . . .	80
6.2. Prueba 2: Traza completa generada por un pedido . . . . .	80
6.3. Prueba 3: Predicción sobre la calidad del producto . . . . .	85
<b>V Conclusiones</b>	<b>87</b>
<b>7. Conclusiones</b>	<b>89</b>
7.1. Trabajos futuros . . . . .	90

7.2. Conclusión . . . . .	90
<b>Referencias bibliográficas</b>	<b>91</b>

# Índice de figuras

1.1. Representación de la distribución geográfica de población con intolerancia a la lactosa. Los tonos azules indican individuos lactasa persistente y los tonos rojos lactasa no persistente (Modificado de Szilagyi and Ishayek, 2018) . . . . .	7
4.1. Diagrama BPMN - Fase proveedores . . . . .	29
4.2. Diagrama BPMN - Fase transporte a fabrica . . . . .	29
4.3. Diagrama BPMN - Fase fabrica . . . . .	30
4.4. Diagrama BPMN - Fase transporte a almacén . . . . .	30
4.5. Diagrama BPMN - Fase almacén . . . . .	31
4.6. Modelo UML . . . . .	32
4.7. Diagrama de componentes . . . . .	34
5.1. Etapas de la cadena . . . . .	38
5.2. PCB Fritzing . . . . .	45
5.3. Esquemático Fritzing . . . . .	46
5.4. Pedido JLCPCB . . . . .	46
5.5. Representación tridimensional de la carcasa para el dispositivo IoT . . . . .	47
5.6. Diagrama de ejecución ESP32 Dual Core . . . . .	47
5.7. Node red - API Rest . . . . .	50
5.8. Node red - Ejemplo nodo inyección . . . . .	51
5.9. Node red - Ejemplo nodo función . . . . .	51
5.10. Node red - Nodos disponibles . . . . .	52
5.11. Node red - Proceso AA . . . . .	53

## ÍNDICE DE FIGURAS

5.12 ■ode red - Proceso AB . . . . .	54
5.13 ■ógica Proceso B . . . . .	55
5.14 ■ode red - Proceso B . . . . .	55
5.15 ■ode red - Proceso CA . . . . .	56
5.16 ■ode red - Proceso CB . . . . .	57
5.17 ■ode red - Proceso CE . . . . .	58
5.18 ■ode red - Proceso CF . . . . .	58
5.19 ■ode red - Proceso CF . . . . .	58
5.20 ■ode red - Proceso EA . . . . .	59
5.21 ■ode red - Proceso EB . . . . .	59
5.22 ■atos guardados en MongoDB . . . . .	60
5.23 ■atos guardados en MongoDB . . . . .	63
5.24 ■atos guardados en MongoDB . . . . .	63
5.25 ■structura Transformer . . . . .	67
5.26 ■structura Estimator . . . . .	67
5.27 ■est Service [Internet]. Arquitecturajava.com. 2022 [cited 2022 Nov 27]. Available from: <a href="https://www.arquitecturajava.com/wp-content/uploads/springrestservi">https://www.arquitecturajava.com/wp-content/uploads/springrestservi</a>	
5.28 ■erencia de anotaciones . . . . .	70
5.29 ■odelos utilizados en Java . . . . .	71
5.30 ■gin y vista principal . . . . .	74
5.31 ■ista de la aplicación . . . . .	75
5.32 ■ista de la aplicación . . . . .	76
6.1. Prototipo dispositivo IoT . . . . .	80
6.2. Prueba 3 . . . . .	85

# Índice de cuadros

5.1. Tabla características técnicas SIM800L . . . . .	41
5.2. Tabla características técnicas ESP32 . . . . .	42
5.3. Tabla características técnicas NEO-6M . . . . .	43
5.4. Tabla características técnicas DHT21 . . . . .	43
5.5. Tabla rango de datos correctos . . . . .	57

---

## Parte I

---

### Introducción

---

## Contexto

**E**n esta sección daremos una breve introducción en el contexto del problema, ademas del estado actual de las tecnologías y los posibles riesgos que pueda entrañar el proyecto.

### 1.1 Introducción

En 2020 se estima que casi el 70 % de los gerentes españoles afirman que han tenido que tomar medidas contra el COVID-19 por medio de alguna transformación digital según un informe de Salesforce <sup>1</sup>. Una situación global que está adelantando lo inevitable, la digitalización de las empresas. Esta transformación a la que llamamos Industria 4.0, tiene como objetivos la interconexión de los procesos mediante dispositivos de internet de las cosas, control de información en tiempo real y ayuda en la toma de decisiones con analítica de datos, entre mas tecnologías <sup>2</sup>. Esto es posible por la gran cantidad de datos que se recogen durante los procesos, ya sea entornos industriales como una fabrica, en ventas con los sistemas de recomendación para los usuarios o incluso el trafico de nuestros equipos en forma de logs, este nuevo modelo esta centrado en los datos como principal recurso en la organización para la toma de decisiones. Este proyecto esta formado por tres pilares fundamentales: Internet de las cosas (IoT), Big Data e Inteligencia Artificial. Aunque bien es cierto que no son los únicos pues campos como la robótica y la computación en la nube cumplen un papel fundamental en este cambio de paradigma.

Al igual que obtenemos datos para controlar los procesos y los productos, un mantenimiento correctivo y preventivo de la maquinaria de las empresas equivale a prolongar la vida útil de estas además de cumplir más correctamente con la seguridad requerida. Una falta de mantenimiento, por lo tanto, nos puede acarrear en no cumplir con los plazos previstos por la rotura de alguna maquina o incluso puede incurrir en un riesgo para los trabajadores o el medio ambiente. El mantenimiento reactivo y correctivo por tanto seria idóneo para anticiparse a las posibles roturas o problemas que se puedan dar. Bajo esta perspectiva, la incorporación de técnicas predictivas en el mantenimiento ha aumentado significativamente en las últimas décadas debido al creciente uso de tecnologías relacionadas con estas capacidades y al beneficio aportado con el aumento de la vida útil de los activos; la reducción de piezas de reemplazo, los costos de mano de obra, energía entre otros. Asimismo, como resultado del desarrollo de nuevas estrategias de mantenimiento <sup>3</sup>.

Durante los siguientes capítulos veremos como una empresa, en nuestro caso láctea, quiere digitalizar la misma y tomando como referencia las tres tecnologías mencionadas, veremos qué pueden aportar cada una dentro de este proyecto. Para ello recogeremos datos mediante sensores IoT, realizaremos el diseño de los dispositivos tanto hardware como software mostrando los componentes empleados y el código desarrollado para su funcionamiento, estos dispositivos son los encargados de generar los preciados datos que serán un pilar fundamental para el desarrollo de cualquier proyecto de aprendizaje automático. Las herramientas utilizadas para el proceso de análisis y transformación de datos, esta siendo actualmente muy demandada por las empresas <sup>4</sup>, en concreto nos referimos a Spark que es un es un

framework de computación en clúster open-source utilizado para el almacenamiento, procesamiento y análisis de grandes volúmenes de datos. Por último mediante técnicas de machine learning buscaremos patrones con el fin de mejorar la producción de la empresa o mejorar la calidad de los productos de una forma más eficiente a la usada tradicionalmente, detectando los productos que han tenido fallos en sus procesos y han quedado defectuosos. Con toda la información disponible, resulta interesante realizar una trazabilidad de por qué sitios, en qué momento y con qué valores han ido evolucionando nuestros productos a lo largo de toda la cadena de suministro.

El desarrollo de esta propuesta se realizará a una empresa ficticia cuya actividad principal es la distribución y fabricación de leche. Escogido este sector por el aumento de intolerancias de este tipo en la población y la creciente preocupación con los hábitos alimenticios<sup>5</sup>. Ya que se estima que aproximadamente dos tercios de la población adulta mundial tiene esta sintomatología.

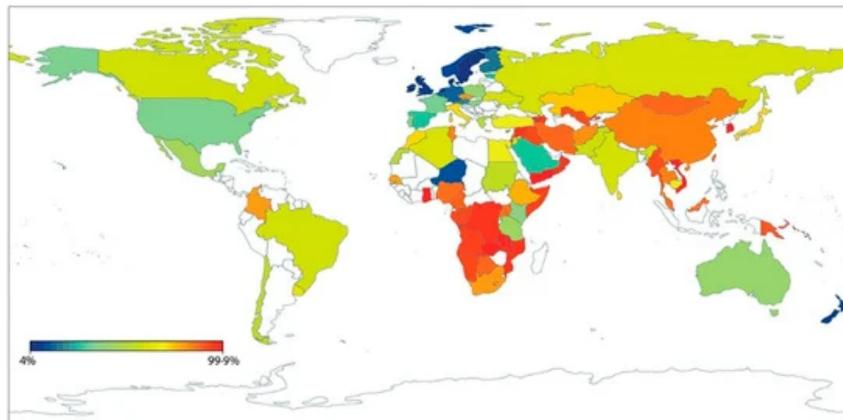


Figura 1.1: Representación de la distribución geográfica de población con intolerancia a la lactosa. Los tonos azules indican individuos lactasa persistente y los tonos rojos lactasa no persistente (Modificado de Szilagyi and Ishayek, 2018).

Como podemos ver en la figura §1.1, la intolerancia a la lactosa es un trastorno que está fuertemente influenciado dependiente del grupo étnico, sexo y edad (Fassio et al., 2018). Fundamentalmente se da en individuos asiáticos, afectando entre el 90% y 100% de su población. Africanos y árabes también se ven afectados con un 80% de la población, mientras que solo afecta al 15% de la población europea. Dentro de la población europea la predominancia es mayor en los países mediterráneos, situando a España en la zona intermedia, ya que el 35% de la población adulta española padece intolerancia a la lactosa. Es menor en los países del norte de Europa, afectando entre el 0 y el 10% (Fernández, 2017)<sup>5</sup>. Esto plantea un escenario perfecto para la resolución del problema para averiguar patrones en la calidad y una trazabilidad logística del producto.

## 1.2 Estado del arte

La conceptualización de la industria 4.0 no está definida oficialmente por ninguna entidad, mas bien se trata de una definición consensuada por las organizaciones que se mueven dentro de este ámbito, podríamos definirla como la unión de la maquinaria y el software que permite la comunicación entre estos para trabajar en red. Sistemas capaces de interactuar con los humanos y capaz de integrarse en entornos de producción, sistemas de almacenamiento o logística. Esto nos permite controlar, predecir y gestionar las organización de una manera inalcanzable hasta día de hoy a causa de las limitaciones tecnológicas<sup>2</sup>.

No obstante se encuentra en proceso de desarrollo, sus beneficios nos permiten adaptar hoy grandes cambios como la digitalización de la producción, automatización de los procesos, maquinaria inteligente, avances en la robótica, toma de decisiones en tiempo real y por supuesto un gran numero de aplicaciones que todavía están por desarrollar dado la corta edad de esta tecnología. Aunque no existe una definición exacta al respecto, el Internet de las cosas, el computo en la nube, analítica de datos y la inteligencia artificial parecen ser los pilares fundamentales de la industria 4.0<sup>2</sup>.

Los avances de estas áreas en los últimos años han dado lugar a progresos en la inteligencia artificial. Un buen ejemplo de esto es la visión por ordenador que permite a los dispositivos reconocer y entender el mundo de forma similar a la que concebimos los humanos. Utiliza principalmente algoritmos para identificar y analizar imágenes, actualmente las funciones más utilizadas son el reconocimiento facial y el reconocimiento de imágenes<sup>6</sup>. Otro campo que se ha visto fuertemente impulsado estos últimos años es el procesamiento de lenguaje natural (NLP) capaz de comprender y analizar el lenguaje humano<sup>7</sup>, algo que lleva en desarrollo desde los años 50 con el primer gran auge de la inteligencia artificial, pero que por aquel entonces se trataba de una colección de reglas como si de un libro de preguntas y respuestas se tratara. Hoy en día los modelos son capaces de resumir textos, generar diálogos, traducirlos; gracia a la capacidad de analizar morfológicamente y semanticamente las entradas de texto que se les proporciona.

La aceleración por GPU es un nuevo marco de trabajo que nos habrá las puertas estas tecnologías, redes neuronales que antes tardábamos meses en entrenar son ahora capaces de realizar este entrenamiento en días gracias a la potentes funciones de computo en paralelo que nos ofrecen<sup>8</sup>. Visión y lenguaje son capacidades multidisciplinarias capaces de integrarse en cualquier área que lo requiera o necesite ya que como nos ocurre a los humanos, estas dos capacidades nos sirven como catalizador al mundo que nos rodea. Algo similar pasa con el Internet de las cosas y el Big Data, son tantas sus aplicaciones que hoy en día no solo puede ser útil en un área concreta. Incluso dentro de la propia industria vislumbra ya el "Internet

industrial de las cosas” o IIoT como resultado del avance realizado en monitoreo ambiental, agricultura, industrias de la construcción, hogares o ciudades inteligentes, gestión de desastres, sistemas asistidos por energía solar, redes inteligentes, tecnología robótica, atención médica, industrias automotrices, sistemas de respuesta a emergencias o sistemas de gestión de la cadena de suministro.

### 1.3 Riesgos del proyecto

En esta sección listamos los riesgos a los que nuestro proyecto puede enfrentarse en su etapa de desarrollo. Estos pueden influir en un incremento de tiempos de planificación y aumento de los costes del proyecto.

- Generación de datos válidos para el proyecto. Puede llegar a darse el problema ya que todos los datos generados a lo largo de la cadena de suministro serán generados aleatoriamente y siempre dentro de unos rangos lógicos y tomando como referencia algunos catálogos de maquinaria industrial láctea, pero lo determinante es que tenemos que estar seguros de generar la suficiente información para que generar modelos predictivos válidos.
- Dispositivo IoT defectuoso o averiado. Es posible que algunos de los módulos hardware utilizados durante el proyecto no funcionen correctamente o que sufra averías inesperadas.
- Dispositivo IoT complejo. Sobre todo nos referimos al envío de datos vía GPRS y sin posibilidad de enviar estos de forma local ya que no disponemos de red Wifi para ello.
- Inexperiencia en las tecnologías. Falta de experiencia del desarrollador del proyecto en gran parte de las tecnologías que se usarán en el proyecto. Esto puede ocasionar una demora en la finalización del proyecto.
- Tecnologías adaptadas al problema. Es muy probable que durante el desarrollo del proyecto algunas tecnologías deben de ser testeadas antes de implementar de forma definitiva para comprobar si el funcionamiento que ofrece se adapta al problema.

## Objetivos

**E**n esta capítulo explicaremos por que hemos elegido este proyecto y cuales son los objetivos del mismo.

### 2.1 Motivación

El gran auge que están teniendo este tipo de tecnologías basadas en datos es cada vez más habitual en el ámbito profesional por la cantidad de aplicaciones que se le puede dar. Ya sea para el estudio de tratamientos médicos o temas tan dispares como el sistemas de recomendación en plataformas digitales.

El tema principal de este proyecto, ademas de mostrar como se componen e interconectan todas estas tecnologías. Es dar a la empresa una herramienta para tomar mejores decisiones, esto nos sera útil por ejemplo cuando queremos tratar un producto y este ha tenido algún error en la cadena. Resulta intuitivo dejar de invertir recursos ya sean materiales o económicos en un producto que aunque complete la cadena, no cumplirá los requisitos de calidad que la empresa estipula. Por esto mismo la visualización en tiempo real y al toma de decisiones son el punto clave de este proyecto, no se generaran mas ingresos por la venta de mas productos sino por la optimización de los ya existentes

### 2.2 Lista de objetivos

Por todo lo mencionado anteriormente, el objetivo de este trabajo es presentar una solución real a una empresa que tras comprobar los beneficios que aportan todas estas tecnologías quiere incluir en su entorno de trabajo. Podríamos especificar los siguientes puntos en el desarrollo del proyecto.

- Diseño Hardware y Software de un dispositivo IoT. Este contará con sensor de temperatura y humedad, un módulo GPS y un módulo GPRS para enviar los datos sin necesidad de estar conectado a una red local.
- Creación del flujo de datos para simular una cadena de suministro real.
- Usar técnicas de Inteligencia Artificial para generar un modelo de datos que sea capaz de predecir si un producto cumple con los requisitos de calidad establecidos.
- Implementar la trazabilidad de productos dentro de nuestro sistema

De forma paralela nos ayudará:

- Estructurar un proyecto en diferentes etapas desde un enfoque más profesional.
- Uso de librerías muy utilizadas en el mundo del Big Data o analítica de datos.

- Planificación temporal del proyecto.
- Todas las habilidades que se irán adquiriendo de forma adicional, como son el manejo de bases de datos no relacionales o la utilización de Python como lenguaje encargado de tratar con los datos.

---

## Parte II

---

### Análisis y Diseño

---

## Análisis de requisitos

**E**sta sección explicara la arquitectura que tiene el proyecto, así como las fases que lo componen. También se enumeraran los distintos requisitos que debe tener el proyecto para cumplirse con éxito.

### 3.1 Requisitos de información

- **RI-01:** La información guardada en el proceso de identificación de bidones debe tener la siguiente estructura:
  - Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Trabajador: String
  - Proveedor: String
  - Tanques de leche: Array String
- **RI-02:** La información guardada en el proceso de preparar el pedido para la fase de primera transporte debe tener la siguiente estructura:
  - Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Trabajador: String
  - Proveedor: String
  - Lote del pedido: String
  - Tanques de leche: Array String
- **RI-03:** La información guardada en el proceso de transporte a la fabrica debe tener la siguiente estructura:
  - Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Trabajador: String
  - Transporte: String
  - Lote del pedido: String
  - Temperatura: Double
  - Humedad: Double

- Latitud: Double
  - Longitud: Double
  - Tanques de leche: Array String
- **RI-04:** La información guardada en el proceso de identificar los tanques de leche en la fabrica debe tener la siguiente estructura:
- Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Trabajador: String
  - Lote del pedido: String
  - Tanques de leche: Array String
- **RI-05:** La información guardada en el proceso de acondicionamiento en tanques debe tener la siguiente estructura:
- Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Identificador de tanque: String
  - Temperatura: Double
  - Presión: Double
- **RI-06:** La información guardada en el proceso de pasteurización debe tener la siguiente estructura:
- Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Identificador de tanque: String
  - Temperatura: Double
  - Presión: Double
- **RI-07:** La información guardada en el proceso de homogeneización debe tener la siguiente estructura:

- Identificador de trazabilidad: String
- Etapa: String
- Identificado del proceso: String
- Timestamp: Long
- Identificador de tanque: String
- Temperatura: Double
- Presión: Double

■ **RI-08:** La información guardada en el proceso de embotellado debe tener la siguiente estructura:

- Identificador de trazabilidad: String
- Etapa: String
- Identificado del proceso: String
- Timestamp: Long
- Identificador de tanque: String
- Temperatura: Double
- Presión: Double

■ **RI-09:** La información guardada en el proceso de preparar el pedido para la segunda fase de transporte debe tener la siguiente estructura:

- Identificador de trazabilidad: String
- Etapa: String
- Identificado del proceso: String
- Timestamp: Long
- Trabajador: String
- Identificador lote de pallet: String
- Pallets: Array String

■ **RI-10:** La información guardada el segundo proceso de transporte hasta el almacén debe tener la siguiente estructura:

- Identificador de trazabilidad: String
- Etapa: String
- Identificado del proceso: String
- Timestamp: Long

- Trabajador: String
  - Transporte: String
  - Lote del pedido: String
  - Temperatura: Double
  - Humedad: Double
  - Latitud: Double
  - Longitud: Double
- **RI-11:** La información guardada en el proceso de identificar los pallets en el almacén debe tener la siguiente estructura:
- Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Trabajador: String
  - Identificador lote de pallet: String
  - Pallets: Array String
- **RI-12:** La información guardada en el proceso de almacenamiento debe tener la siguiente estructura:
- Identificador de trazabilidad: String
  - Etapa: String
  - Identificado del proceso: String
  - Timestamp: Long
  - Trabajador: String
  - Identificador lote de pallet: String
  - Almacenamiento: Objeto Almacenamiento
- **RI-13:** La información guardada en el Objeto Almacenamiento debe tener la siguiente estructura:
- Identificador del pallet: String
  - Identificador de pasillo: String
  - Número de estantería: Integer

### 3.2 Requisitos funcionales

- **RF-01:** Una vez recibido un pedido, el proveedor identificará los bidones donde se introducirá la leche para comenzar la cadena
- **RF-02:** Los bidones identificados en el proceso AA, se verifican y se agrupan en un único lote antes de ser enviados a la fase transporte 1
- **RF-03:** Los camiones encargados de transportar los tanques de leche o pallets deben enviar información sobre su estado en tiempo real.
- **RF-04:** Se verificarán e identificarán los bidones de leche que han llegado a la fábrica para posteriormente ser procesados
- **RF-05:** Para una mejor conservación del producto, se introducirán los tanques recibidos por el proveedor en unos tanques especiales para acondicionar y mantener mejor el producto
- **RF-06:** El proceso de pasteurización debe simular un tratamiento térmico de intensidad media, que consiste en la aplicación de temperaturas de entre 60 - 85 °C y tiempos de entre 15 segundos hasta 30 minutos, en función de la intensidad que requiera el producto
- **RF-07:** El proceso de homogeneización consiste obtener una fase líquida homogénea, en cuanto al contenido en grasa se refiere. Es por esto que la presión sera mucho mayor que en otros procesos.
- **RF-08:** El proceso de embotellado consiste en almacenar bajo condiciones asépticas en envases estériles y herméticos, es necesario controlar la temperatura y humedad en este proceso también.
- **RF-09:** En la ultima etapa de la fabricación, los envases serán agrupados en palets según el bidón de origen y serán identificados y preparados para su envío antes de ser enviados a la segunda fase transporte
- **RF-10:** Se debe verificar e identificar los palets que han llegado al almacén para posteriormente ser almacenados
- **RF-11:** Se guardará la información acerca del almacenamiento de los palets recibidos
- **RF-12:** El sistema tiene que ser capaz de crear datos realistas, dado que vamos a hacer predicciones sobre una batería de datos de prueba, los datos que generemos para la fase de entrenamiento tienen que estar acotados dentro de un rango lógico de valores que han sido estudiados previamente como los que generaría normalmente el proceso de la industria láctea

- **RF-13:** El sistema tiene que ser capaz de hacer predicciones sobre la calidad de un producto, uno de los objetivos principales es poder mostrar esta predicción en la aplicación móvil
- **RF-14:** El sistema tiene que ser capaz de generar la trazabilidad de un producto, uno de los objetivos principales es poder mostrar la trazabilidad de un producto en la aplicación móvil dado el número de lote de un envase de leche
- **RF-15:** El dispositivo desarrollado debe recoger datos de temperatura, humedad y GPS
- **RF-16:** El dispositivo desarrollado debe enviar datos fuera de la red local donde se encuentre, dado que se trata de un dispositivo instalado en un transporte, no disponemos de red wifi y por tanto es un punto necesario que sea capaz de enviar datos en un ámbito global (WAN)
- **RF-17:** Proceso de login para un usuario, la aplicación debe disponer de un proceso de login para el acceso de usuarios concretos
- **RF-18:** La aplicación debe disponer de una pestaña donde se consulte la trazabilidad de un producto concreto

### 3.3 Requisitos no funcionales

- **NRF-01:** Las tecnologías utilizadas deben de ser acorde con lo que actualmente se usa en el mercado, mostrarnos un caso real
  - Spark
  - MongoDB
  - Node-red
- **NRF-02:** Las tecnologías utilizadas deben de ser acorde con lo que actualmente se usa en el mercado
  - Angular
  - Ionic
- **NRF-03:** Guardado de datos en MongoDB, todos los datos generados y procesados en el caso de la trazabilidad de un producto tienen que ser guardados en una base datos mongoDB
- **NRF-04:** Guardado de datos en la nube, todos los datos guardados tienen que estar disponibles en una nube privada para poder hacer uso de ellos fuera de la red local.

- **NRF-05:** La aplicación debe ser multiplataforma, poder hacer uso en dispositivos IOS y Android

## Diseño del sistema

**E**sta sección se encarga de explicar los casos de uso, procesos, modelo de datos y arquitectura del sistema

## 4.1 Casos de uso

En esta sección explicaremos las interacciones que surge entre el sistema y los diferentes elementos del proyecto, esto nos ayudara a conseguir objetivos específicos. Los siguientes esquemas de casos de uso muestra la interacción de los usuarios con el sistema.

### CU-01: Acceder a la aplicación

- Precondición del proceso: El usuario debe de tener la aplicación instalada.
- Descripción: El sistema debe comportarse como se describe cuando el usuario quiere acceder a la aplicación
- Secuencia normal:
  1. El usuario introduce sus credenciales
  2. El sistema verifica que ese usuario este registrado y cumple las credenciales
  3. El usuario accede a la interfaz principal de la aplicación
- Postcondición: Se permite el acceso a la aplicación
- Excepciones:
  - 2. Si el usuario no se encuentra en el sistema o sus credenciales no coinciden
    - 2.1. El sistema informa al usuario que las credenciales no son correctas
    - 2.2. Termina el caso de uso
- Comentarios: No hay un proceso de registro por lo que será el administrador del sistema el encargado de inscribir los nuevos usuarios

### CU-02: Consultar la trazabilidad de un productos

- Precondicion del proceso: El usuario debe de tener acceso a la aplicación
- Descripción: El sistema debe comportarse como se describe cuando el usuario consulta el estado de un producto con la aplicación
- Secuencia normal:
  1. Accedemos a la aplicación
    - 1.1. Se realiza el caso de uso "Acceder a la aplicación"

2. El usuario introduce el código del producto en la barra de búsqueda o escanea el código de barras del mismo
  3. El sistema solicita todos los procesos relacionados con el código introducido
  4. El sistema solicita las predicciones sobre estos procesos
  5. La aplicación recibe los datos y los visualiza
- Postcondición: El usuario podrá ver la traza completa del producto seleccionado
  - Excepciones:
    - 3. Si el sistema no encuentra procesos con el identificador proporcionado
      - 2.1. El sistema informa al usuario que hay procesos disponibles para ese identificador
      - 2.2. Termina el caso de uso
  - Comentarios: Los procesos que se hayan determinado como erróneos, se muestran con un contorno rojo mientras que aquellos que han tenido un proceso normal se marcan en verde.

#### **CU-03:** Simular la generación completa de una cadena

- Precondición del proceso: Debemos especificar a la aplicación Node-red cuales es la conexión con la base de datos
- Descripción: El usuario puede generar una cadena desde los proveedores hasta el almacenamiento haciendo uso de la herramienta Node-red
- Secuencia normal:
  1. El usuario acciona el botón "Hacer pedido" situado en la fase de proveedores
  2. En cada proceso el sistema almacenará los datos correspondiente al mismo
- Postcondición: Se guarda en la base de datos Mongo la traza generada
- Excepciones:
  - 2. Podremos modificar en la herramienta que valores máximos y mínimos tiene un proceso en particular

#### **CU-04:** Generar predicciones

- Precondición del proceso: No son necesarias condiciones previas
- Descripción: Podremos indicar al sistema que queremos realizar una predicción de los datos guardados en la base de datos "TFGCollectionToProcess" y se guarda en "TFGCollectionProcessed".
- Secuencia normal:
  1. El usuario ejecuta la aplicación Spark realizada en el proyecto con el parámetro "predict"
  2. El sistema carga en la aplicación los modelos que nos permitirán hacer predicciones
  3. El sistema lee de la base de datos todos los procesos que aun no han sido tratados y han sido directamente guardado en esta
  4. El sistema proceso los datos y los transforma en un formato que puede ser consumido por el modelo
  5. El sistema aplica el modelo a los datos transformados
  6. El sistema guarda la predicción de los procesos
- Postcondición: El usuario tiene acceso a la trazabilidad de ese producto y visualizar su pronostico

## 4.2 Procesos

Para una mejor visualización del proceso, se ha dividido todo el modelo de negocio en cinco diagramas, cada uno representando una etapa del proyecto.

### 4.2.1 Etapa de Proveedores

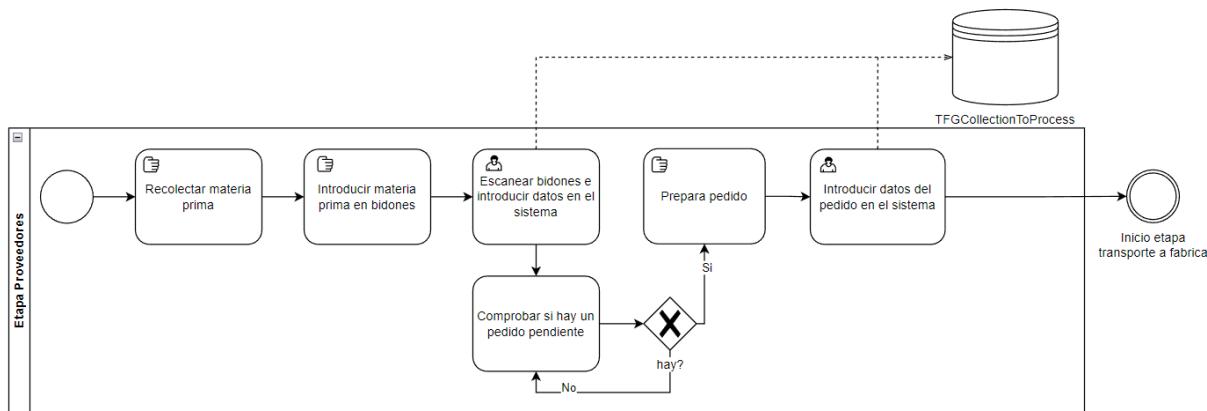


Figura 4.1: Diagrama BPMN - Fase proveedores

### 4.2.2 Etapa de transporte a la fabrica

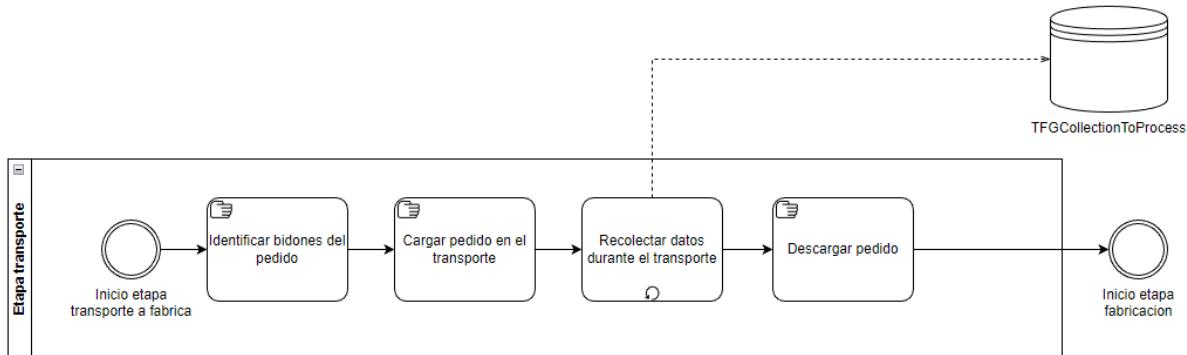


Figura 4.2: Diagrama BPMN - Fase transporte a fabrica

#### 4.2.3 Etapa de fabricación

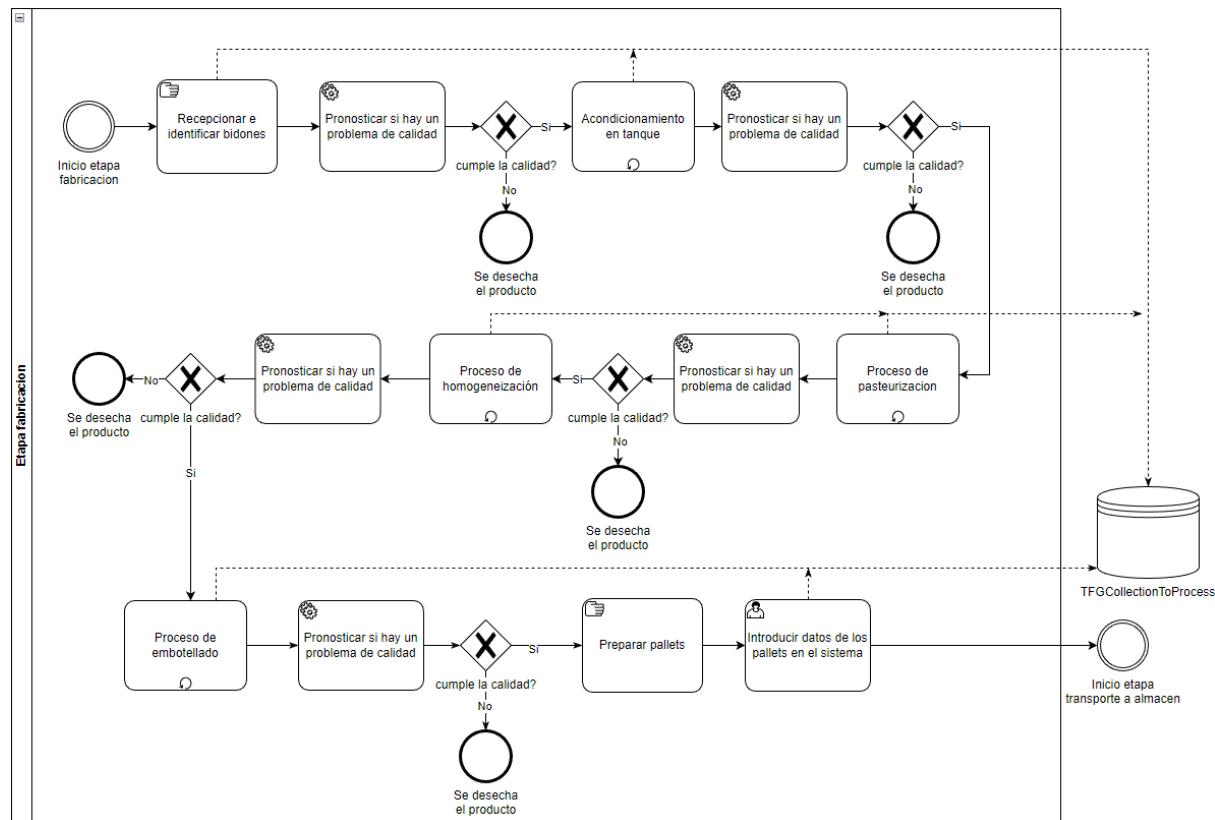


Figura 4.3: Diagrama BPMN - Fase fabrica

#### 4.2.4 Etapa de transporte al almacén

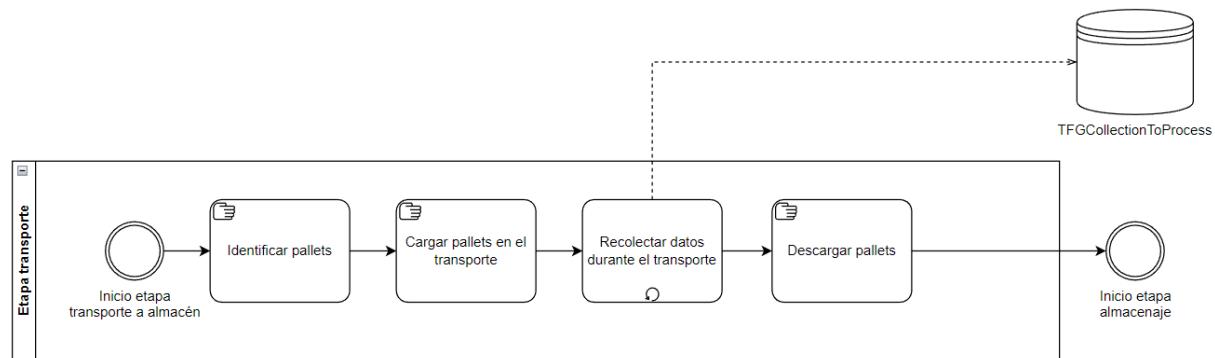


Figura 4.4: Diagrama BPMN - Fase transporte a almacén

#### 4.2.5 Etapa de almacenamiento

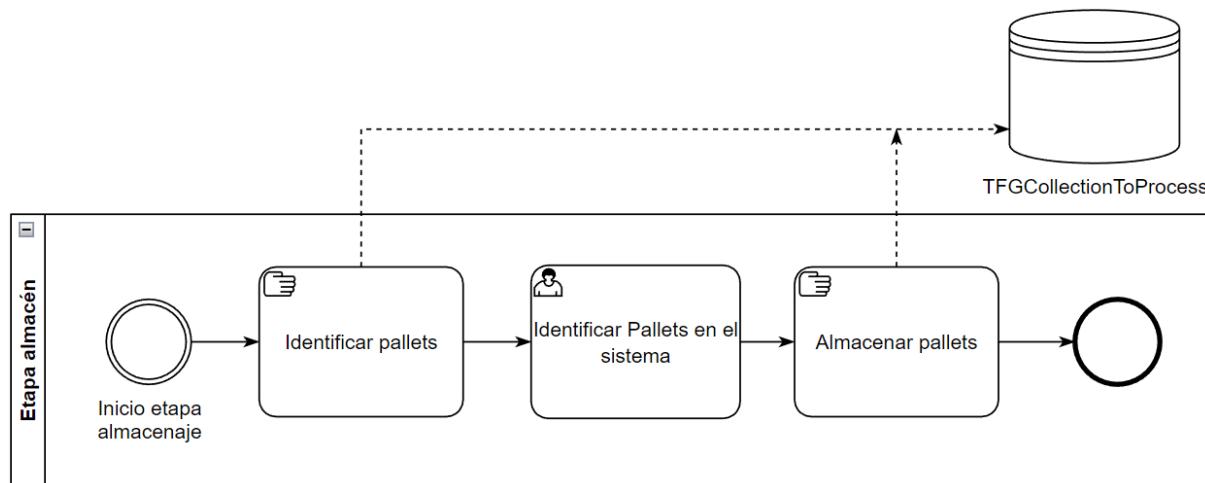


Figura 4.5: Diagrama BPMN - Fase almacén

### 4.3 Modelo de datos

En el siguiente diagrama podemos ver como se relacionan los datos entre si, esta formado por lo siguiente elementos:

- **UserAuthModel**: Es el modelo encargado de guardar las credenciales de un usuario y hacer único a este en el sistema
- **GenericTask**: Este modelo se encarga de almacenar todos los parámetro disponibles para cada tarea
- **ProcessedTask**: Este modelo guarda los datos de las predicciones hechas por Spark
- **StorageModel**: Es la clase encargada de guardar los datos en la fase de almacenamiento

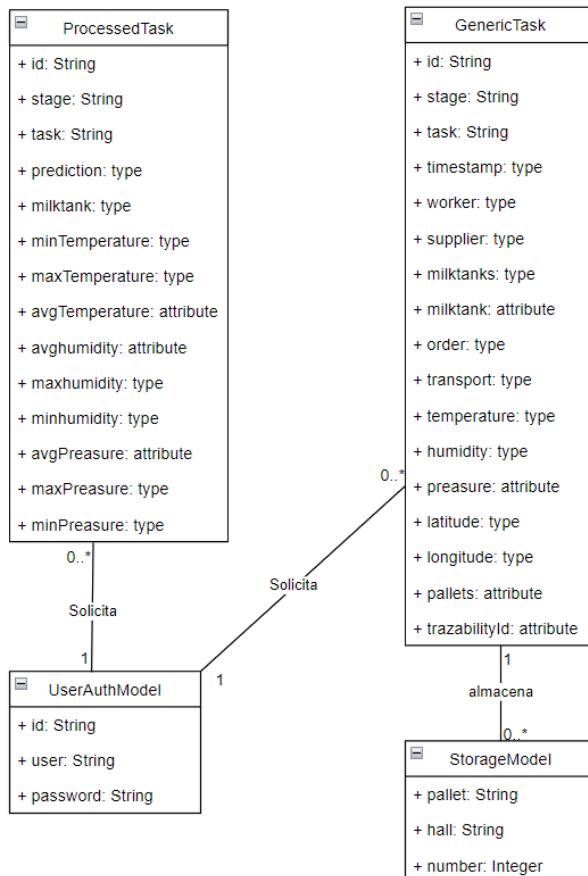


Figura 4.6: Modelo UML

#### 4.4 Arquitectura lógica del sistema

En el siguiente diagrama mostramos que estructura tiene el sistema, ademas de como se conectan los distintos componentes del proyecto

- **Controller:** Es el componente encargado de administrar todas las peticiones realizadas al servidor, en ella se encuentran los objetos que nos permiten usar los métodos de cada clase instanciada
- **Service:** se encarga de verificar la identidad del usuario que quiere acceder a la applicacion
- **MongoRepository:** Interfaz del repositorio MongoDB utilizado.
- **UsersRepository, GenericTaskRepository, ProcessedTaskRepository:** Estas clases heredan de MongoRepository y nos permiten realizar acciones en la base de datos especificada
- **SparkService:** Es el encargado de la comunicacion entre el servidor y la aplicacion Spark
- **SparkAction:** Este componente ejecuta la aplicación Spark que le indiquemos
- **Node red:** Es el componente encargado de generar los datos que utilizamos en el proyecto

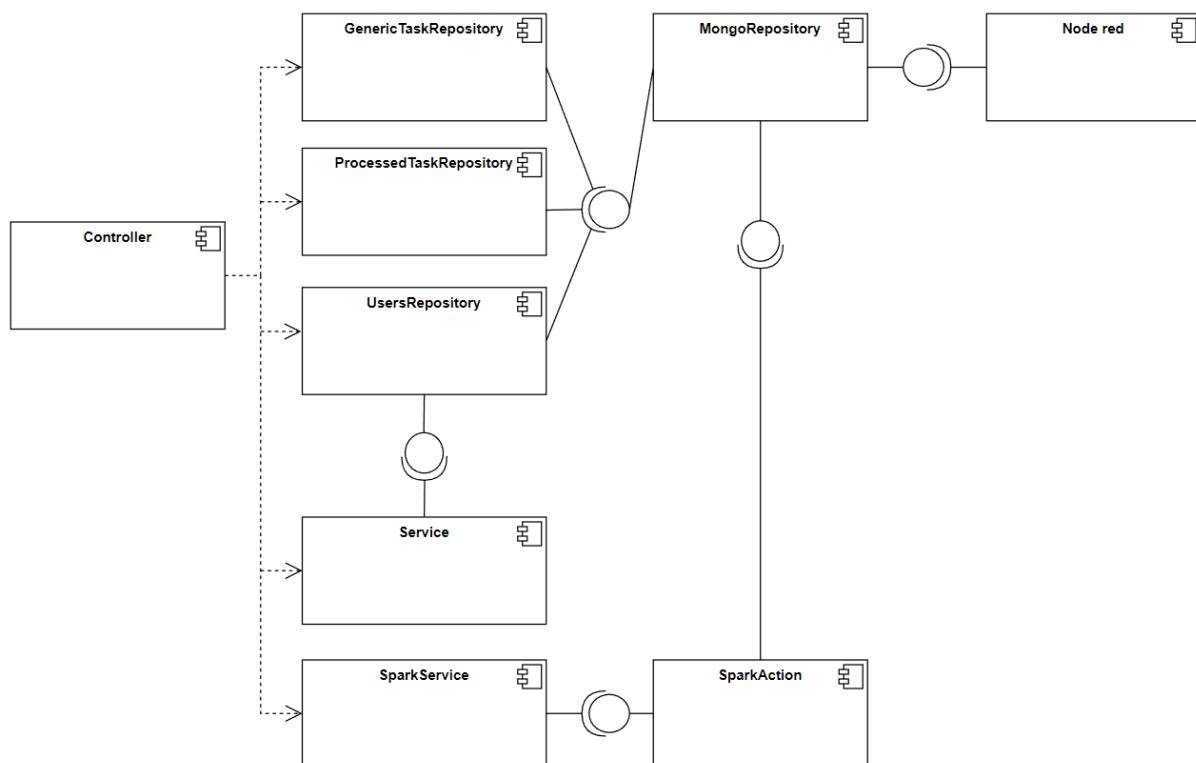


Figura 4.7: Diagrama de componentes

---

## **Parte III**

---

### **Desarrollo del proyecto**

---

## Desarrollo e implementacion

**E**n este capítulo implementaremos todas las tecnologías anteriormente mencionadas. En el explicaremos los detalles de la implementación y como interconectar todas estas tecnologías.

## 5.1 Arquitectura del sistema

A continuación, explicaremos a alto nivel cuales son cada una de las partes que intervienen en el proceso además de explicar cuál es su función dentro de la arquitectura del proyecto. Como podemos ver en el diagrama §5.1, el sistema está compuesto por 5 etapas y los datos generados en cada fase del proceso serán especificados en los requisitos de información.

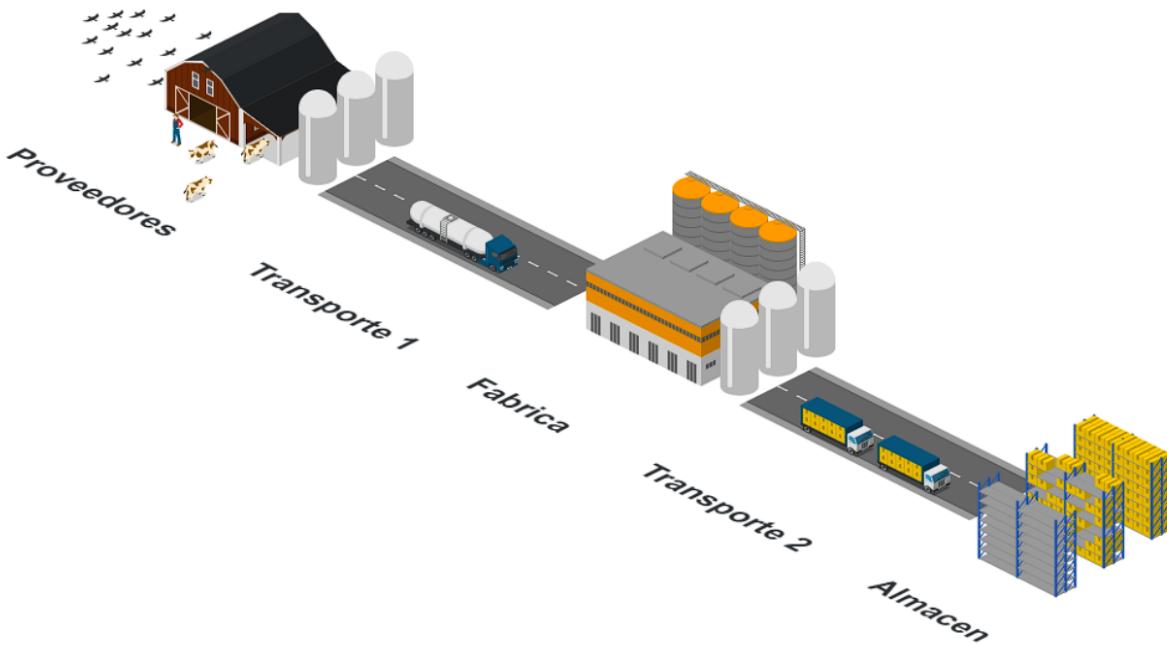


Figura 5.1: Etapas de la cadena

Las fases que componen el sistema son las siguientes:

- Fase proveedores. En primer lugar se identificarán los bidones de leche que han sido recolectados y seguidamente se procederá a prepararlos y asignarles un número de pedido para su posterior envío.
- Fase transporte 1 (Proveedores a fábrica). Es el proceso encargado de transportar la materia prima a la fábrica para ser procesada.
- Fase fabricación. Una de las fases más importantes del proceso y que generan más datos. Cuando el pedido llega a la fábrica, el primer paso es identificar los bidones con el identificador que se le asigno en la fase de transporte anterior. Una vez registrados en el sistema, se comenzara a procesar la leche con los procesos de almacenaje en el tanque, pasteurización, homogeneización y embotellado. Ademas se agruparan en pallets que serán enviados al almacén.

- Fase transporte 2 (Fábrica - almacén). Este proceso representa el transporte de los productos ya listos para su consumo a un punto de almacenaje o venta.
- Fase almacenamiento. Es la ultima fase de la cadena y se encargara de almacenar los pallets recibidos en un punto concreto del almacén.

## 5.2 Tecnologías y herramientas utilizadas

Para desarrollar cada parte del proyecto se ha hecho uso de múltiples tecnologías, cada una nos ofrece características que han sido sometidas a un estudio previo para comprobar que se adapta a nuestras necesidades. Fundamentalmente se ha tenido en cuenta que las herramientas utilizadas sean de software libre o que el servicio que ofrece es económico dado las necesidades del proyecto.

- **PlatformIO** es un IDE (Integrated Development Environment) de código abierto para la programación en C/C++ y orientado a hardware. Esto nos da un conjunto de herramientas dentro de su interfaz gráfica para facilitar el desarrollo al programador. Una de estas ventajas es el fichero platformio.ini que nos permite centralizar las configuraciones del proyecto y la placa física utilizada, esto nos permite descargar e instalar todas las dependencias necesarias incluso si trabajamos en distintos entornos de desarrollo.
- **Java** es un lenguaje de programación orientado a objetos y que se ha desarrollado bajo el framework **Spring**. Un framework basado en HTTP y servlets, que nos proporciona herramientas para la personalización de aplicaciones web y servicios web REST.
- **Apache Spark** es un framework de programación para procesamiento de datos distribuidos diseñado para ser rápido y de propósito general. Esta desarrollado en el marco de Apache, lo que garantiza un modelo de desarrollo basado en el consenso y la cooperación, en una licencia de software abierta. Consta de diferentes APIs que permite tratar el dato en todas las etapas de su ciclo de vida. Algunas librerías que nos han sido incluidas en este proyecto nos permite realizar alguna función como representar grafos o procesamiento de datos en tiempo real pero las APIs utilizadas en este son las siguientes:

**Spark Core** es un motor distribuido de uso general para procesar datos.

**Spark SQL** es el módulo de Spark que permite utilizar datos estructurados.

**MLLib** es la biblioteca escalable de aprendizaje automático de Spark.

- **Node-red**, desarrollado por IBM, ayuda a combinar el hardware, su propia API y otros servicios en línea de manera intuitiva y eficiente. Es un editor

de flujo accesible desde el navegador. Tiene compatibilidad con los grandes sistemas operativos como IOS, Windows y Linux. Esta formado por dos componentes como se explica a continuación.

**Nodos:** Los nodos están escritos en Javascript aunque muchos tienen funcionalidades predeterminadas. Se puede instalar fácilmente desde la biblioteca de Node-Red.

**Flujos:** Los diagramas de flujo de Node-Red se crean integrando varios nodos que se configuran y almacenan mediante JSON.

- **MongoDB** es un sistema de administración de bases de datos NoSQL. Almacena datos como documentos similares a JSON con esquemas dinámicos (el formato se llama BSON). MongoDB tiene su enfoque orientado a cuatro cosas: flexibilidad, potencia, velocidad y facilidad de uso. Admite servidores replicados e indexación y ofrece controladores para múltiples lenguajes de programación.
- **Ionic** este framework de código abierto, nos permite generar aplicaciones para dispositivos móviles que nos ofrece numerosas herramientas y facilidades para desarrollar interfaces móviles con apariencia nativa. Basado en tecnologías web (HTML, CSS y JS) nos permite desarrollar aplicaciones para iOS nativo, Android y la web, desde una única base de código.
- **Fritzing** es una aplicación de software libre que te permite crear y diseñar circuitos. Muy flexible e intuitiva, forma un ecosistema que permite a los usuarios documentar y compartir sus prototipos y crear esquemas de circuitos impresos.

### 5.3 Dispositivo IoT

El desarrollo de este proyecto comienza por el diseño e implementación del dispositivo IoT y que es el encargado de generar los datos que usaremos. Hasta ahora hemos estructurado teóricamente una empresa que produce datos, esta estructura esta totalmente adaptada a las necesidades del proyecto y podría cambiar si estas los requieres, ademas en un entorno real cada punto de la cadena esta formado por dispositivos distintos a nivel hardware. Es lógico pensar que si nos encontramos en la fase de fabricación de un producto los parámetros que nos resultan mas interesantes de recoger son aquellos que influyen directamente en la calidad producto o por ejemplo si nos encontráramos en la fase de transporte desde los proveedores a la fabrica, quizás atributos como la localización GPS serian mas interesantes.

Para este proyecto se ha optado por implementar uno de los dispositivos mas completos de toda la cadena y simular el resto de dispositivos como se verá en la

siguiente sección **Generación y procesamiento de los datos**. Este dispositivo se encuentra en el transporte desde los proveedores hasta la fábrica o desde la fábrica hacia el almacén y podríamos imaginarlo como una caja negra parecida a la de los aviones que además de recoger la información, la envía. Decimos que es uno de los más interesantes a desarrollar porque no tenemos una red local de Internet a la que conectarnos para enviar los datos como es el caso de una fábrica o un almacén donde tenemos un router al que conectarnos. Para solventar esto, haremos uso de un módulo con una tarjeta SIM para enviar los datos vía GPRS como son las redes 3G o 4G que suelen utilizarse en nuestros dispositivos móviles.

Los sensores y módulos utilizados en este dispositivo son los siguientes:

- **SIM800L** este módulo nos permite enviar y recibir datos, SMS y llamadas. Además, soporta la red GSM/GPRS cuatri-banda que son estándares a nivel mundial: 850, 900, 1800 y 1900 MHz. En la mayoría de los países de Europa, Asia, Australia, Medio Oriente y África, se emplean las bandas de 900-1800 MHz. En la parte trasera del módulo tenemos un socket SIM en la que introduciremos una tarjeta MicroSIM activada y con cobertura 3G o 4G, esta almacena de manera segura tu número de teléfono, así como las claves de acceso de un usuario concreto en una operadora de telefonía.

Características técnicas	
<b>Tensión de alimentación</b>	3,8 V - 4,2 V
<b>Consumo de energía</b>	modo de suspensión <2,0 mA modo inactivo <7,0 mA
<b>Transmisión GSM</b>	350 mA, (pico): 2000 mA
<b>Tamaño del módulo</b>	25 x 23 cm
<b>Frecuencias admitidas</b>	850/950/1800/1900 MHz

Cuadro 5.1: Tabla características técnicas SIM800L

- **ESP32** es un potente microcontrolador SoC con Wi-Fi 802.11 b/g/n integrado, modo Bluetooth versión 4.2 y variedad de periféricos. Es una actualización del conocido ESP8266 cuya diferencia principal es la agregación de dos núcleos sincronizados de entre 80 hasta 240 MHz lo que es fundamental para nuestro proyecto ya que nos permitirá recoger datos de temperatura, humedad, GPS y simultáneamente en otro núcleo poder estar enviando estos datos. Comparado con su predecesor, además de estas características, también amplía el número de pines GPIO de 17 a 36, el número de canales PWM (Modulación por ancho de pulso) por 16 y está equipado con 4 MB de memoria flash. Tenemos la posibilidad de utilizar SPI, I2S, I2C, CAN y UART con varios puertos

de cada uno. El dispositivo estándar también incluye sensor de efecto Hall, sensor de temperatura y sensor táctil. El ESP32 también proporciona aceleración de hardware para algoritmos orientados a la encriptación: AES, SHA-2, RSA y criptografía de curva elíptica (ECC) y generador de números aleatorios (RNG). Los aceleradores permiten incrementar la velocidad de operación posibilita encriptado y des-encriptado dinámico. De esta manera podemos proteger activamente el sistema de posibles ataques de hackeo que pretendan obtener el código almacenado.

Características técnicas	
<b>Voltaje de alimentación</b>	5V
<b>Voltaje de entrada / salida</b>	3.3V
<b>Corriente de Funcionamiento min.</b>	500mA
<b>SoC</b>	ESP32-WROOM 32
<b>Frecuencia de Reloj</b>	80MHz / 240MHz
<b>RAM</b>	512kB
<b>Memoria Flash externa</b>	4MB
<b>Pines I / O</b>	34
<b>Interfaces</b>	SPI, I2C, I2S, CAN, UART
<b>Protocolos Wi-Fi</b>	802.11 b/g/n (802.11n hasta 150 Mbps)
<b>Frecuencia Wi-Fi</b>	2.4 GHz - 2.5 GHz
<b>Bluetooth</b>	V4.2 - BLE y Bluetooth clásico
<b>Antena inalámbrica</b>	PCB
<b>Dimensiones</b>	56x28x13mm

Cuadro 5.2: Tabla características técnicas ESP32

- **NEO-6M** es un módulo que nos permitirá recibir información precisa sobre la geo-localización del dispositivo, gracias a una antena cerámica y una memoria EEPROM para guardar los datos y la configuración del módulo, también incluye una pequeña batería para conservar la información y no perder esta. Sus entradas y salidas están compuestas por cuatro pines: TX, RX, VCC y GND, el voltaje de operación es de 3.3V - 5V, pero la lógica del puerto serial (Tx, Rx) es de 3.3V. Los satélites GPS cumplen dos órbitas completas de formas precisa cada 24 horas, cada uno transmite una señal única y parámetros orbitales que permiten al módulo GPS decodificar y calcular la ubicación precisa del satélite. Los receptores GPS utilizan esta información y trilateración para calcular la ubicación del dispositivo. Esencialmente, el receptor GPS mide la distancia a cada satélite por el tiempo que tarda en recibir una señal emitida.

Con la distancia a tres satélites, el receptor puede determinar la posición del dispositivo aunque la precisión aumentará a mas numero de satélites.

Características técnicas	
<b>Voltaje de alimentación</b>	3-5 V
<b>Interfaz</b>	Serial UART 5V
<b>Baud rate</b>	9600bps
<b>Frecuencia de refresco</b>	5Hz
<b>Dimensiones</b>	25mm x 35mm

Cuadro 5.3: Tabla características técnicas NEO-6M

- **DHT21** este sensor digital nos permitirá recoger los datos de temperatura y humedad con buena precisión.

Características técnicas	
<b>Voltaje de alimentación</b>	3.5 - 5.5V
<b>Consumo de energía</b>	Serial UART 5V
<b>Tiempo de sensado</b>	2s
<i>Temperatura</i>	
<b>Rango</b>	-40 hasta 80°C
<b>Precisión</b>	+ - 0.5°C
<i>Humedad</i>	
<b>Rango</b>	0 a 100 % RH
<b>Precisión</b>	+ - 3 %

Cuadro 5.4: Tabla características técnicas DHT21

También serán necesarios otros componentes para la analógica del dispositivo, cumplirán un importante papel a la hora de proteger a este de picos de intensidad, uno de ellos es un regulador de tensión ajustable que nos da una salida de 3A, necesario para hacer funcionar correctamente el modulo **SIM800L** teniendo este picos de hasta 2A. Ademas como recibiremos la energía de una batería seria aconsejable eliminar el ruido eléctrico generado con un condensador que suavice la señal.

Con la herramientas **Fritzing** podremos diseñar una PCB (Placa de circuito impreso) que conecte todos los sensores y módulos deseados. **Fritzing** cuenta con 4 pestañas principales.

- **Protoboard** en esta pestaña tenemos una vista central donde podemos ir colocando los componentes que se seleccionan en la columna derecha, tiene soporte para muchos fabricantes como Arduino, SparkFun e Intel entre muchos otros, aun que también tenemos la posibilidad de descargar de forma externa los componentes ya que al ser un programa de código abierto bastante utilizado en el diseño de prototipos, tiene mucha comunidad y los componentes mas populares se encuentran sin problema.
- **Esquemático** muestra una representación gráfica del circuito electrónico, en el que se utilizan las huellas de los componentes que intervienen en él y sus conexiones.
- **PCB** una de las vistas mas importantes de toda la herramienta si no es la que mas, es esta pestaña podremos distribuir los componentes en la placa y unirlos mediante pistas, la herramienta nos permite incluir distintas capas de conexión si fuese necesario.
- **Código** en esta pestaña podemos incluir código para ser cargado en una placa física en el caso de necesitar realizar alguna prueba.

El primer paso sera buscar la plantilla de los componentes que vamos a utilizar en nuestro proyecto. Para ello podemos buscarlos en la columna "componentes" desde la propia herramienta si se trata de componentes genéricos como las resistencias o condensadores o muy populares como las plantillas Arduino. También podemos buscarlos en foros públicos de la propia pagina de **Fritzing** o en repositorios Github de usuarios que los comparten en repositorios privados.

En esta primera vista podemos ver la posición que tendrán los componentes en la placa impresa, las pistas son de 24 micras de espesor y tiene un tamaño de 105.8 x 58.8 mm. Lo complicado en este punto del diseño es ser capaz de integrar todas las conexiones necesarias en una única capa de silicio ya que si necesitamos incluir mas capas, incrementara el coste de fabricación cuando la solicitemos.

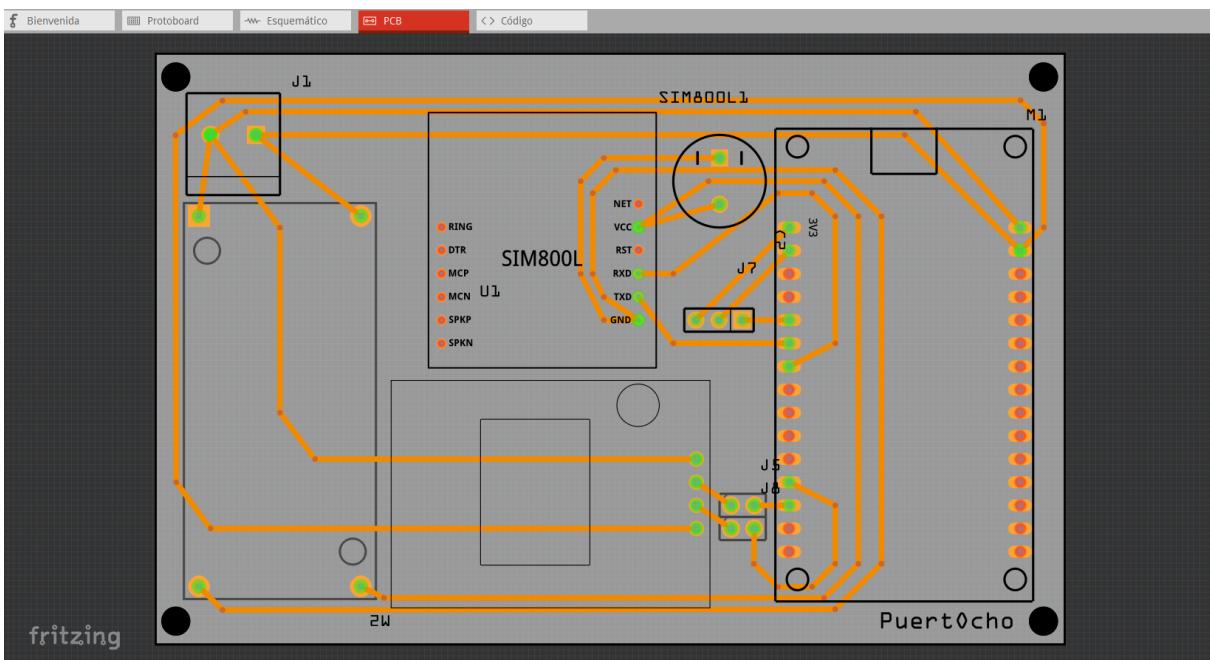


Figura 5.2: PCB Fritzing

El esquematico nos da una visión simplificada del circuito con símbolos normalizados y a diferencia de la vista anterior la posición de los componentes no es importante. Como podemos ver en la imagen, hay cuatro cuadros en rojo que corresponden con las siguientes entradas y salidas físicas del dispositivo.

- **1 y 2:** Se han agregado dos Jumpers a los pares **GPIO01/TX0 (ESP32)-RX (NEO-6M)** y **GPIO03/RX0 (ESP32) - TX (NEO-6M)** para facilitar el testeo de la placa.
- **3:** Dos pines donde podremos darle corriente a nuestro circuito con un transformador de 5V y 3A.
- **4:** Tres pines donde conectaremos nuestro sensor **DHT21**.

Una vez tenemos todas conexiones hechas y nuestro diseño listo para funcionar tendremos que exportar los archivos Gerber, estos archivos son documentos vectoriales que contienen toda la información acerca de la placa impresa como la descripción de las conexiones, pistas, vías y pastillas de una PCB ademas de las instrucciones para fabricarla.

El fabricante por el que hemos optado es la compañía asiática de placas impresas **JLCPCB**, que se especializa en prototipos de PCB rápidos y producción de PCB de lotes pequeños y medianos. Tiene una web intuitiva a la hora de subir los documentos y te informa bastante bien sobre el estado del pedido.

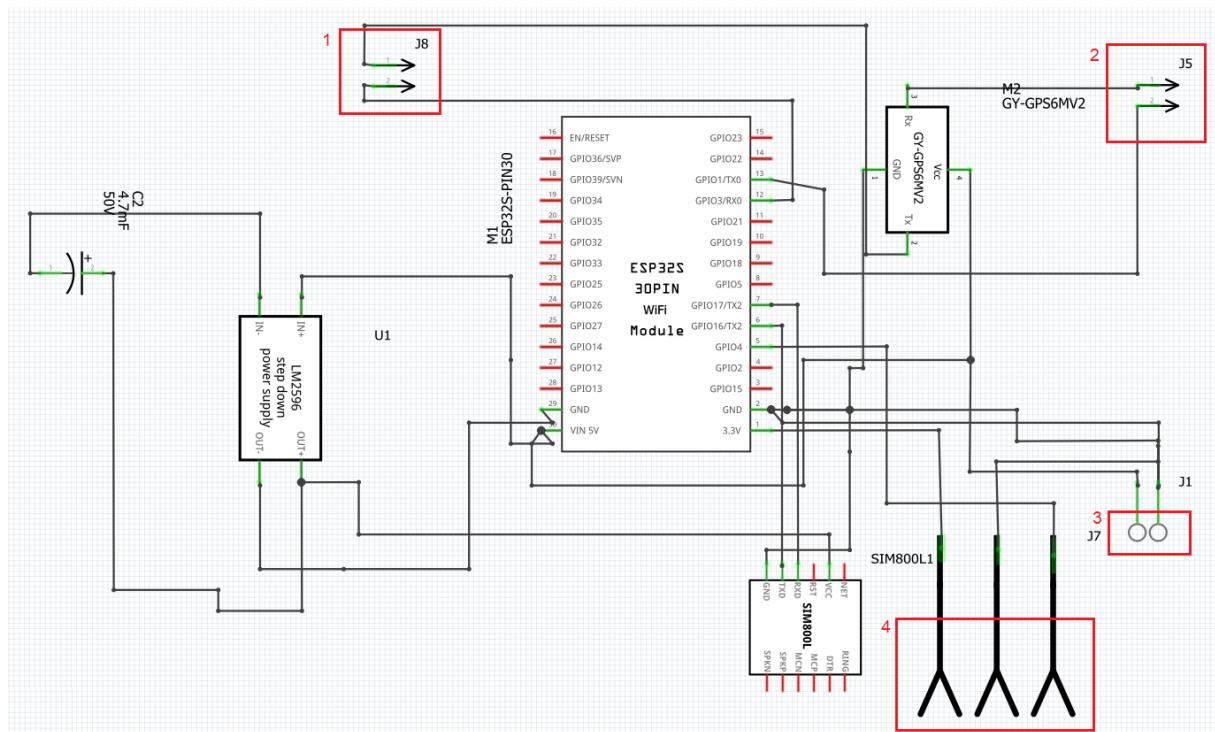


Figura 5.3: Esquemático Fritzing



PCB Prototype	PCB_Y1	Merchandise Total: €1.64
	Reviewing	Shipping Charge: €4.90
Order #: <a href="#">Order #</a>	File review will be completed in 10-60 mins when business hours.	Order Total: €6.54
Build Time: 3 days		
5 pcs €1.64		
<a href="#">Product Details</a>		

Figura 5.4: Pedido JLCPCB

El siguiente paso será diseñar una envoltura para proteger los componentes e imprimirla en una impresora de resina 3D, aun que esto es un paso opcional que no influye en el funcionamiento del proceso.

Con esto tenemos todo el hardware necesario para empezar a recoger información por lo que es momento de ver como podemos programar el microcontrolador para realizar todo el proceso de recolección y envío de datos. El flujo que seguiremos sera el representado en la figura §5.6.

Como comentamos anteriormente el micro-controlador dispone de dos núcleos por lo que podemos realizar dos acciones simultáneamente, en nuestro caso el hi-

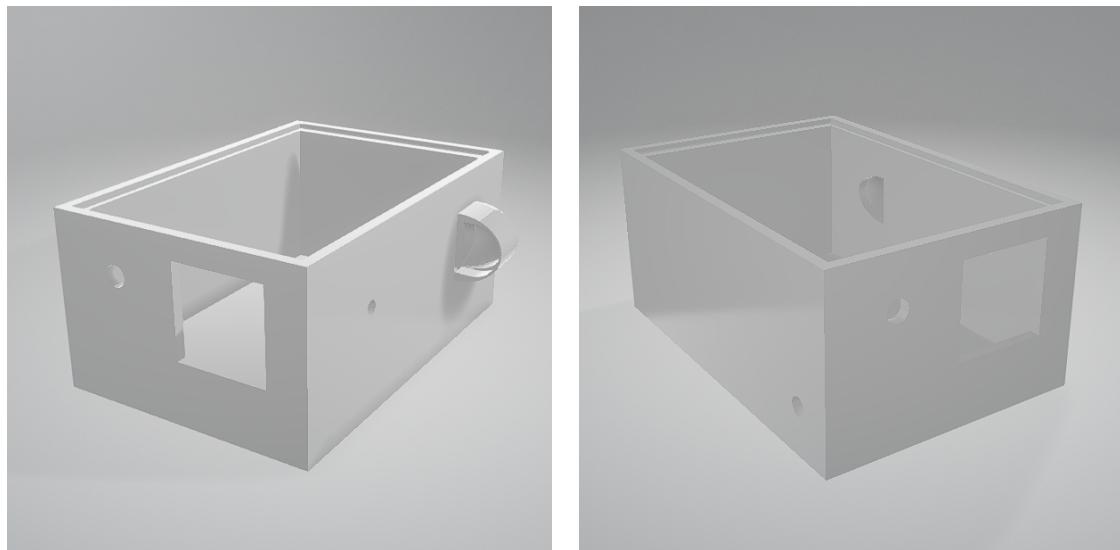


Figura 5.5: Representación tridimensional de la carcasa para el dispositivo IoT

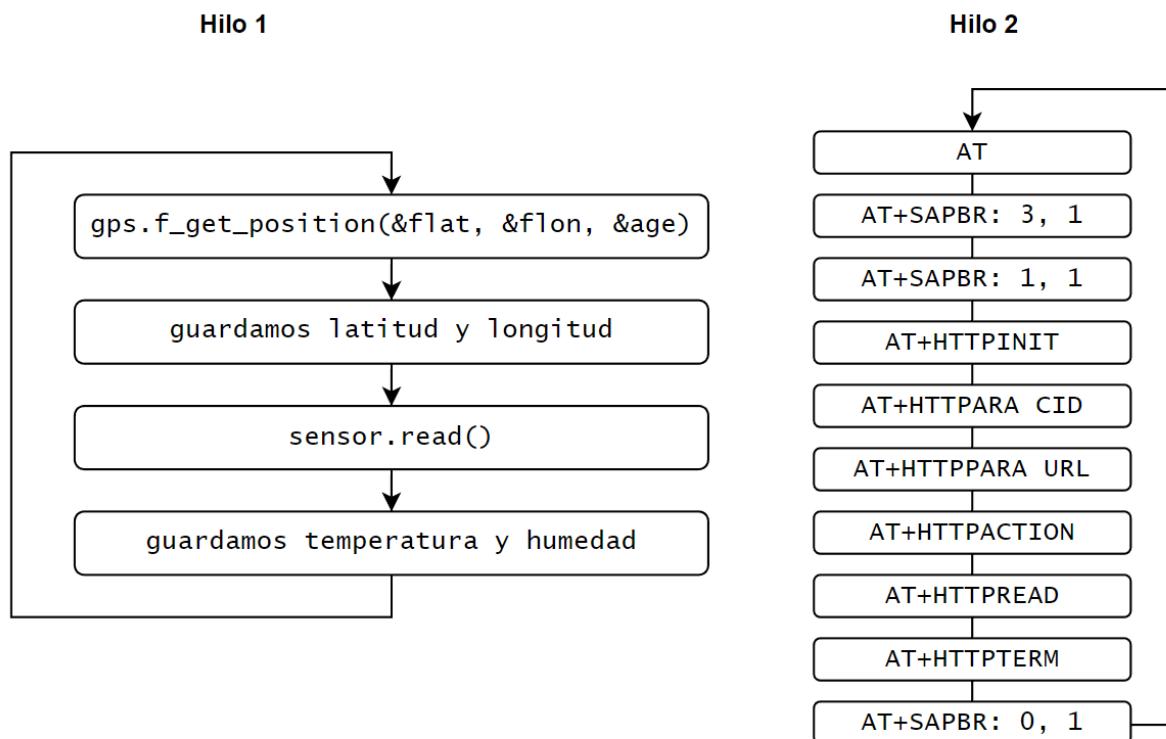


Figura 5.6: Diagrama de ejecución ESP32 Dual Core

lo de ejecución numero uno será utilizado para tratar los datos de posición que recojamos con el modulo GPS **NEO-M6** y humedad y temperatura con el sensor **DHT21**. El hilo de ejecución dos sera el encargado de enviar los datos con el modulo **SIM800L** y mediante comandos AT nos permitirá enviar datos mediante de una

petición HTTP GET.

Para obtener los datos desde los sensores hemos utilizado las librerías **TinyGPS** y **Dhtnew** que de forma sencilla nos permiten transformar la señal recogida en valores numéricos. En el caso de **Dhtnew** solo es necesario hacer uso de las funciones **getHumidity** y **getTemperature** para guardas los valores recogidos en variables. **TinyGPS** nos provee de la función **f\_get\_position** que nos permite guardar la información recogida en los punteros que se le pasa como argumento (&flat y &flon) ademas esta librería también nos permite recoger datos como la altitud, la orientación cardinal o la velocidad en distintas unidades internacionales.

---

```
#include <TinyGPS.h>
#include <dhtnew.h>

TinyGPS gps;
DHTNEW sensor(4);

gps.f_get_position(&flat, &flon, &age);

longitude = String(flon, 6);
latitude = String(flat, 6);

humidity = String(sensor.getHumidity(), 1);
temperature = String(sensor.getTemperature(), 1);

float falt = gps.f_altitude(); // +/- Altitud en metros
float fc = gps.f_course(); // Orientacion en grados
float fk = gps.f_speed_knots(); // Velocidad en nudos
float fmpm = gps.f_speed_mph(); // Velocidad en millas/hora
float fmps = gps.f_speed_mps(); // Velocidad en metros /segundo
float fkmph = gps.f_speed_kmph(); // Velocidad en kilometros / hora
```

---

Los comandos AT fueron diseñados originalmente en los años 80 y todavía se usan en la mayoría de los teléfonos inteligentes para obtener funcionalidades de telefonía. El envío de datos si resulta mas complicado por que no disponemos de una librería que nos facilite el proceso de implementación y requiere de la ordenada ejecución de comando AT para que funcione correctamente. El papel de los comandos AT en estos dispositivos se ha ampliado enormemente a través de personalizaciones específicas de los distintos proveedores.

Como vimos en la figura §5.6 el orden en el que se ejecutan los distintos comandos es muy concreto dado que cada uno tiene una función particular de configuración o conexión a la red dentro del proceso de envío de datos. La estructura que se seguirá para enviar cada comando depende de la acción que vayamos a realizar. Si ejecutamos comandos que nos den información acerca de la conexión y no requieran de ningún parámetro extra, entonces usaremos el patrón **AT+XXXX**.

Si por otro lado requerimos modificar un parámetro de la conexión usaremos el patrón **AT+XXXX=YYY**. Para habilitar la comunicación serie inicializamos **softwareSerial** creando un Objeto Tipo SoftwareSerial asignado a los pines 16 y 17 del micro-controlador y estableceremos la velocidad de comunicación en 19200 baudios. El código siguiente nos muestra como es la estructura básica para enviar un comando AT XXXXX.

---

```
void comandoAT_XXXXX() {
    softwareSerial.println("AT+XXXXX");
    delay(1000);
    String inputChannel = "";
    while(softwareSerial.available() != 0){
        inputChannel = softwareSerial.readStringUntil('\n');
        inputChannel.trim();
        Serial.println("Available_data:" + inputChannel);

        if(inputChannel.equals("OK")){
            Serial.print("success_XXXX");
        } if(inputChannel.equals("ERROR")){
            Serial.print("error_XXXX");
        } else {
            Serial.print("Channel_value:" + inputChannel);
        }
        Serial.print("\n");
    }
}
```

---

Algunos de los comandos utilizados no son necesario para el establecimiento de conexión pero si resultan interesantes para obtener información de la red a la que estamos conectados. Para ser mas precisos, los comandos utilizados en este proyecto son los siguientes:

- **AT**: Este comando se utiliza para verificar la comunicación entre el módulo **SIM800L** y el micro-controlador.
- **AT+SAPBR**: Podremos configurar el servicio de datos (Bearer Service) lo que nos permitirá habilitar la transmisión de información entre interfaces de red.
  - **AT+SAPBR= 3, 1, APN, orangeworld**: Establecemos cual sera el punto de acceso establecido por el operador, en nuestro caso orangeworld.
  - **AT+SAPBR= 1, 1**: Nos conectamos con el portador configurado anteriormente.
  - **AT+SAPBR= 0, 1**: Nos permite desactivar el contexto del portador establecido.
- **AT+HTTPINIT**: Nos permite iniciar el servicio HTTP.

- **AT+HTTPPARA:** Nos permite modificar parámetros del servicio HTTP.
  - **AT+HTTPPARA= CID, 1:** Estableceremos el ID que tenemos en la conexión.
  - **AT+HTTPPARA= URL, http://antoniopuerto.ddns.net:1880/transporte1:**  
Esta instrucción nos permite realizar la petición GET que queremos. Los valores se añadirán como parámetros de solicitud.
- **AT+HTTPACTION:** Con esta instrucción seleccionamos el tipo de petición que vamos a realizar.  
**AT+HTTPACTION=0:** 0 para tipos Get y 1 para POST.
- **AT+HTTPREAD:** Lo usamos para leer la respuesta del servidor HTTP.
- **AT+HTTPTERM:** Cerramos sesión con el servidor HTTP.

El framework **PlatformIO** nos facilitara mucho la tarea de compilar y cargar el código en el micro-controlador así como de importar las librerías de los módulos utilizados. En este punto del proyecto, nuestro hardware ya es capaz de enviar los datos por una petición HTTP GET y recibir esta llamada para guardar los datos. Aunque podríamos haber implementado el punto de acceso de la petición GET en el servidor Spring que hemos utilizado para la aplicación, hemos decidido implementarla en Node-red para mostrar también como desplegar un API Rest simple en esta herramienta.



Figura 5.7: Node red - API Rest

## 5.4 Generación y procesamiento de los datos

### 5.4.1 Generación de datos

Con un concepto tangible de lo que es un dispositivo IoT para recolectar y enviar datos. La herramienta que hemos utilizado para esto es **Node-red**, como comentamos en un principio es un punto de unión idóneo para la conexión hardware y software del proyecto. Esto lo haremos con el fin de poder generar datos realistas que nos permitan hacer predicciones sobre el estado del producto. Esta herramienta esta compuesta por dos bloques principales.

En **Node-red** tenemos dos tipos de **nodos**. Los **nodos de inyección** que generar un mensaje sin necesidad de entrada y lanza este al siguiente nodo conectado. Y los **nodos de función** que tienen una entrada y se encarga de realizar alguna tarea.

La figura §5.8 representa la implementación básica de un **nodo de inyección** que envía un mensaje y es recogido por un nodo de depuración.



Figura 5.8: Node red - Ejemplo nodo inyección

La figura §5.9 representa la implementación básica de un **nodo de función** para mostrar en un nodo de depuración el timestamp que es una marca de tiempo estandarizada.

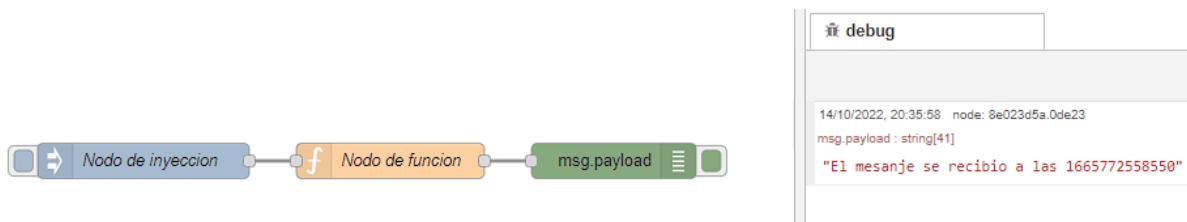


Figura 5.9: Node red - Ejemplo nodo función

Como ya dijimos en la presentación de esta herramienta, disponemos de un amplio abanico de nodos. Unos ya integrados por defecto como se muestra en la figura §5.10 y otros disponibles desde la propia librería de **Node-red** facilitándonos el camino ya que muchas compañías como **MongoDB** desarrollan sus propios módulos y nos ahorrarán tener que implementar todo el proceso desde cero.

Los **flujos** son colecciones de nodos que actúan como uno en el espacio de trabajo. Se pueden usar para reducir la complejidad visual de un flujo o para agrupar un grupo de nodos como un flujo reutilizable que se usa en varios lugares. Nos permite exportar estos flujos en formato JSON, esto lo hace muy versátil a la hora de importarlo en otros flujos proyectos.

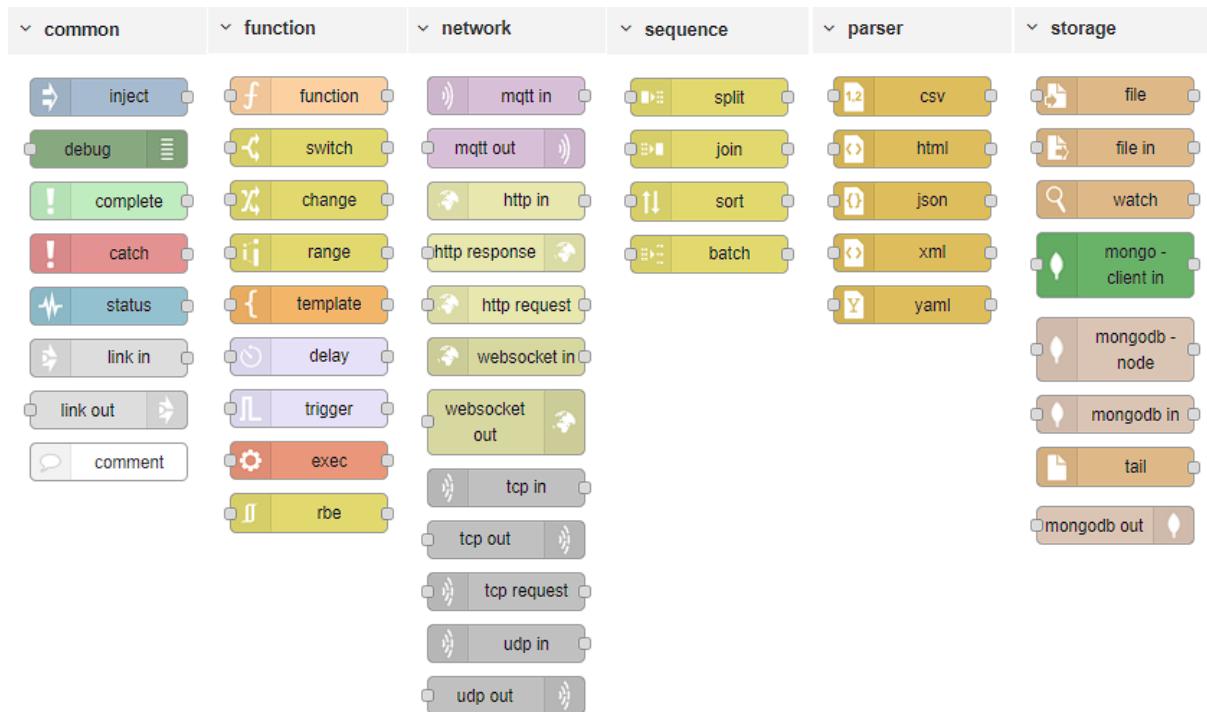


Figura 5.10: Node red - Nodos disponibles

Hemos dividido el proceso en cinco subflujos cada uno dedicado a una etapa de la cadena de suministro.

### Proveedores

Este subfoco genera los datos definidos para los requisitos de información RI-01 y RI-02 que corresponde con los procesos AA y AB.

El requisito **RI-01** corresponde con el proceso **AA**, encargado de identificar los bidones que serán enviados a la fábrica. Comenzamos presionando el nodo de inyección "hacer pedido" que actuara como bandera para indicar a **Node-red** que comience a ejecutar todo el flujo, generando "msg" un objeto por defecto que utiliza Node-red para comunicar los nodos entre si. Este será recibido por el nodo funcional *AA - Identificar Bidones* y con código JavaScript nos permite modificar el mensaje recibido, modificando sus valores o añadiendo parámetros nuevos. Esto nos permite trabajar con los datos recibidos desde el nodo anterior para modificar sus valores y crear una traza como la que crearía un producto en la cadena logística.

Las primeras líneas de código las dedicaremos a la implementación de funciones que realicen una tarea particular dentro del nodo, en nuestro caso se han utilizado para generar cadenas de caracteres aleatorios que usaremos como identificadores en algunos parámetros del proceso.

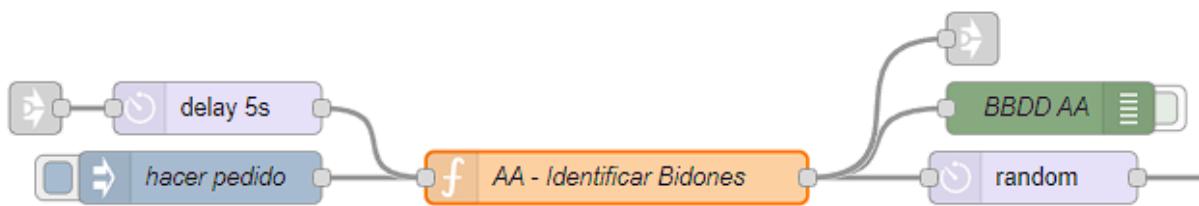


Figura 5.11: Node red - Proceso AA

- **makeid(length)**: Genera un identificador alfanumérico aleatorio de una longitud igual al parámetro *length*.
- **getRandomInt(min, max)**: Genera un numero aleatorio entre los valores *min* y *max*.
- **generateMilkTanks(length)**: Genera una lista de *length* de longitud con identificadores aleatorios.

---

```

makeid(length) {...}
getRandomInt(min, max) {...}
generateMilkTanks(length) {...}

msg = {};
msg.trazabilityId = makeid(12);
msg.stage = 'Proveedor';
msg.task = 'AA';
msg.timestamp = Date.now();
msg.worker = 'Trabajador_' + getRandomInt(0, 4);
msg.supplier = 'Proveedor_' + getRandomInt(0, 2);
msg.milkTanks = generateMilkTanks(4);
return msg;
  
```

---

La salida de este nodo genera un objeto con los valores definidos en el requisito de información **RI-01** y sera guardado en la base de datos Mongo, se vera el contenido del objeto con un nodo de depuración *BBDD AA* y se esperara un intervalo aleatorio de tiempo antes de pasar al siguiente proceso.

El requisito **RI-02** corresponde con los datos generados por el proceso **AB**, encargado de verificar los bidones y prepararlos para continuar a la fase de transporte. La principal diferencia con el nodo anterior es la forma de recibir los datos ya que a partir de este nodo necesitaremos guardar la información de los nodos previos para mantener una continuidad en los datos.

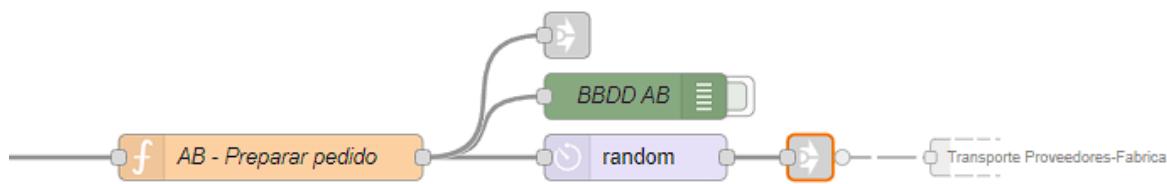


Figura 5.12: Node red - Proceso AB

Al igual que el nodo anterior también guardaremos los datos generados en la base de datos Mongo, lo visualizaremos en el nodo de depuración *BBDD AB* y se esperara un intervalo aleatorio de tiempo antes de pasar al siguiente subflujo.

---

```

var lastMsg = msg;

msg = {};
msg.supplier = lastMsg.supplier;
msg.milkTanks = lastMsg.milkTanks;
msg.trazabilityId = lastMsg.trazabilityId;
...
return msg;
  
```

---

### Transporte 1: Proveedores-Fabrica

Este subfujo genera los datos definidos para el requisito de información RI-03. El requisito **RI-03** corresponde con el proceso **B** encargado del transporte de los productos desde el proveedor a la fabrica.

Como mencionamos antes, este proceso es el encargado de simular los datos que genera el dispositivo IoT. Implementado en la sección anterior aunque también ha sido necesario de desarrollar para mantener una continuidad en los datos ya que como hemos visto hasta ahora, la evolución que van teniendo los mismos a lo largo de todo el flujo es muy dependiente del mensaje recibido por el nodo anterior. La representación de este proceso es distinto a los previos ya que al intervenir mas nodos que en los procesos anteriores, resulta mas claro representarlo en forma de flujo.

Tomando como punto de partida el mensaje del proceso AB, lo primero que haremos sera inicializar los valores constantes para este proceso. Los parámetros **tarea** y **etapa** los estableceremos a "B" y "Transporte 1" respectivamente, mientras que los parámetros **transporte** y **trabajador** se generan aleatoriamente en cada iteración de la cadena como sucedió con **trabajador** y **proveedor** en el proceso anterior. Otros parámetros como **identificador de trazabilidad** y **lote del pedido** se mantendrán con respecto al nodo anterior.

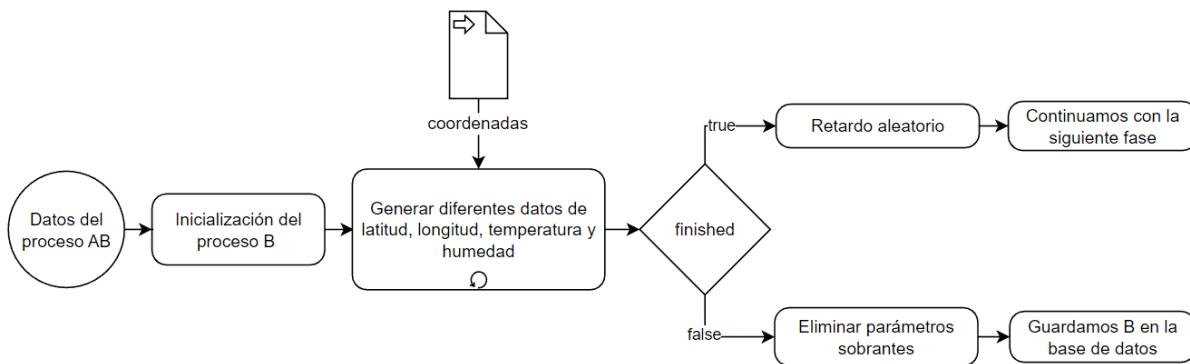


Figura 5.13: Lógica Proceso B

---

```

msg.transport = 'Camion_' + getRandomInt(1, 3);
msg.worker = 'Trabajador_' + getRandomInt(7, 11);
  
```

---

Para poder generar valores de **latitud** y **longitud** realistas, hemos establecido una ruta en Google Maps entre dos puntos del map y hemos extraído 12 puntos equidistantes en dicha ruta para guardar sus valores de **latitud** y **longitud** en un archivo de texto, un punto por linea y separado por una coma. Cuando **Node-red** detecte que el nodo encargado de establecer los valores constantes haya finalizado, empezaremos a leer linea por linea el archivo de texto guardado. En cada iteración guardamos los valores de latitud y longitud leídos y generamos un valor de temperatura y humedad aleatorio dentro de un rango dependiendo si queremos que sea un valor valido o no. Entendiendo como valor valido aquel que se encuentra dentro de un rango de temperatura 0 a 5 grados y una humedad de entre el 60 y 80 por ciento y no valido en caso contrario.



Figura 5.14: Node red - Proceso B

Usando un nodo funcional tipo **Join** podremos unir los datos generados aleatoriamente y los datos constantes generados en el primer nodo. Por cada iteración, comprobaremos si ya hemos leído todos las lineas del archivo de texto que contiene las coordenadas. En el caso de no haber terminado de leer, el parámetro **finished** se mantendrá a falso y se guardara un objeto como el que se muestra en la sección de pruebas 5.2. En cambio si no hay mas datos de coordenadas que leer entonces pasaremos a la siguiente etapa.

## Fabricación

Este subfoco genera los datos definidos para los requisitos de información RI-04, RI-05, RI-06, RI-07, RI-08 y RI-09. Es el encargado de generar la mayoría de datos de nuestro proyecto, lo que cobra gran importancia ya que uno de los objetivos principales es poder realizar predicciones sobre estos para detectar fallos en productos antes de que se incremente su precio al realizar mas procesos.

El requisito **RI-04** corresponde con el proceso **CA** encargado de identificar los tanques que leche provenientes del primer transporte, es la tarea previa al tratamiento del producto.

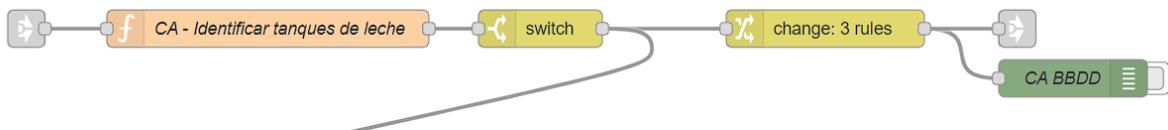


Figura 5.15: Node red - Proceso CA

Establecemos los valores directamente de cada una de sus propiedades, generando aleatoriamente el trabajador y asignando los valores ya definidos en procesos previos como el numero de lote, el identificador de trazabilidad y los identificadores de tanques de leche.

Los requisitos de información **RI-05, RI-06, RI-07 y RI-08** corresponde con los procesos **CB, CC, CD y CE** respectivamente, los datos generados por cada proceso es muy similar y la forma de implementarlos por lo que se explicara conjuntamente.

A diferencia de la etapa anterior, cada tanque de leche sera procesado independientemente. Esto quiere decir que si en cada pedido al proveedor recibimos cuatro tanques de leche, cada uno de ellos tomara valores distintos de presión y temperatura. Se realizaran un total de cinco mediciones por cada tanque de leche y por cada proceso. Para controlar el flujo de datos en **Node-red** se han añadido dos parámetros a todos los procesos de la fabrica. **finished** nos indica si hemos terminado de recoger las mediciones en el proceso que nos encontramos y si se cumple, pasamos al siguiente proceso. **orderFinished** lo utilizaremos para comprobar si todos los tanques de leche del pedido han terminado de ser procesados, si estos es así continuaremos a la siguiente etapa.

Es necesario determinar que rango valores serán correctos en cada proceso de la fabrica como hicimos con la temperatura y humedad en la etapa de transporte. Tomando como referencia el documento "Instalaciones y Maquinaria en la Industria Láctea" del Proyecto Formativo de "Especialista en Quesería" desarrollado por la Consejería de Agricultura, Ganadería, Pesca y Desarrollo Sostenible de la Junta de Andalucía, explica los distintos conceptos relacionados con el diseño de la industria

láctea, así como los equipos e instalaciones que constituyen la tecnología quesera en cada una de los procesos que darán lugar a la producción de queso. Detalla correctamente que valores reales podría tener la materia prima en cada proceso para ser correctos.

Restricciones de los procesos	
<b>CB</b>	
Temperatura	2 a 5 °C
Precisión	0.75 a 1.5 Bares
<b>CC</b>	
Temperatura	70 a 80 °C
Precisión	0.75 a 1.5 Bares
<b>CD</b>	
Temperatura	60 a 80 °C
Precisión	98 a 148 Bares
<b>CE</b>	
Temperatura	15 a 20 °C
Precisión	0.9 a 2.25 Bares

Cuadro 5.5: Tabla rango de datos correctos

La implementación en **Node-red** de los procesos mencionados es muy parecidas en cuanto a código y nodos utilizados. El proceso CB tiene la particularidad de modificar el parámetro **orderFinished**, esto lo haremos si hemos terminado de tomar todas las mediciones de todos los taques de leche recibidos. En el resto de procesos de la fábrica son iguales que la figura §5.17 donde solo se puede modificar el parámetro **finished** y variar los valores de temperatura y presión.

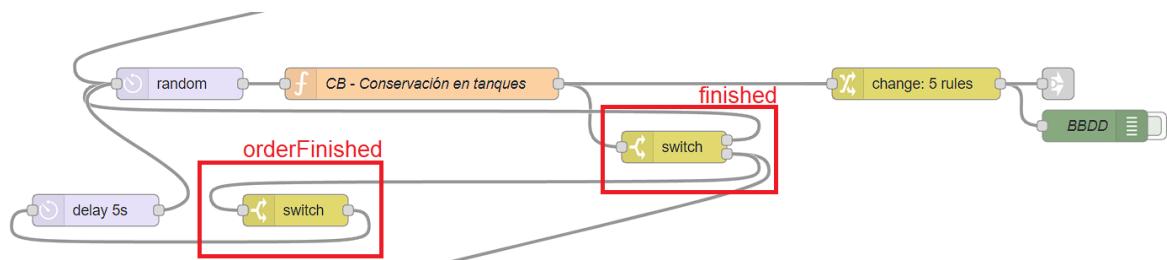


Figura 5.16: Node red - Proceso CB

El requisito **RI-09** corresponde con el proceso **CF** encargado de agrupar y preparar los pallet para ser enviados al almacén en la siguiente etapa. Esto lo haremos siempre y cuando el parámetro **orderFinished** este a **verdadero**, lo que nos indica que la recogida de datos en el proceso de fabricación ha terminado. Cuando esto

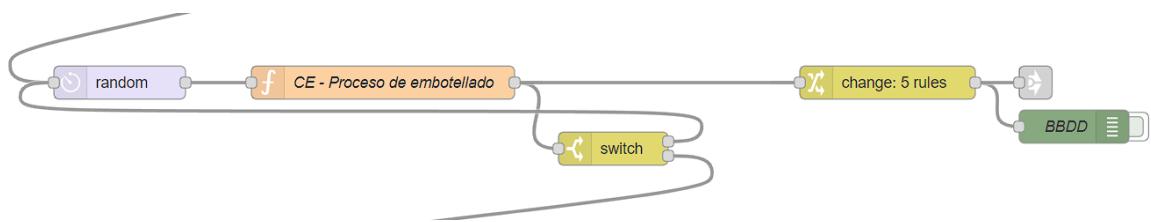


Figura 5.17: Node red - Proceso CE

pase, se guarda el objeto en la base de datos y lo enviaremos a la siguiente etapa de transporte.



Figura 5.18: Node red - Proceso CF

### Transporte Fabrica-Almacén

La información definida para el requisito de información **RI-10**, corresponde con el proceso **D** encargado de transportar los pallets con los productos recién salidos de la fabrica. El objetivo para este proceso es el mismo que para la primera fase de transporte, recogeremos datos de temperatura y humedad y simularemos el recorrido de un camión especificándole coordenadas distintas a las que se le proporcionaron en la primera fase de transporte.

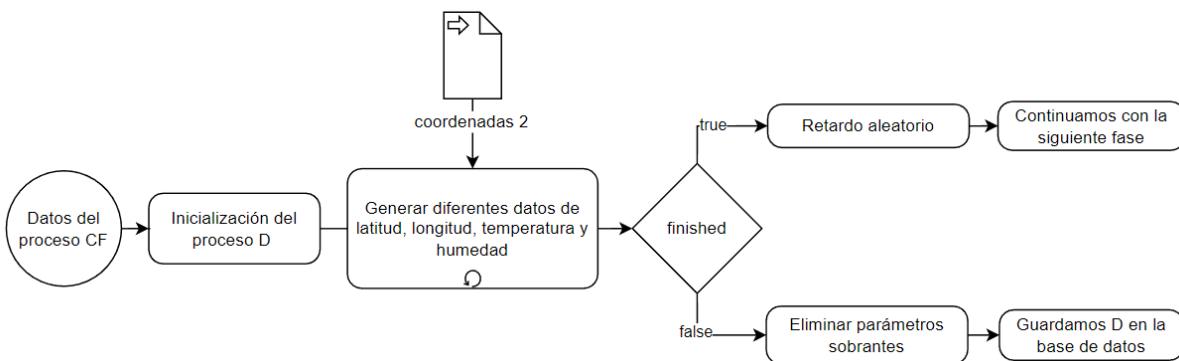


Figura 5.19: Node red - Proceso CF

### Almacén

Este subfoco genera los datos para los requisitos de información **RI-11**, **RI-12** y **RI-13**.

El requisito de informacion RI-11 corresponde con el proceso EA encargado de identificar los pallets que han llegado al almacén. Este proceso genera aleatoria-

mente el parámetro worker, siendo el empleado que identifica el pallet. Y utilizando parámetros de procesos anteriores como order, pallets y trazabilityId. La implementación en **Node-red** para estos dos últimos procesos es muy similar al primero donde identificábamos tanques de leche.



Figura 5.20: Node red - Proceso EA

Los requisitos de información RI-12 y RI-13 corresponde con el proceso CB encargado de almacenar los pallets recién registrados. Esto lo haremos con un objeto de tipo almacenamiento que contendrá en numero de identificación del pallet, el pasillo en el que se guardada y el numero de la estantería. El final de este proceso esta conectado con el principio de la cadena, lo que indicara a **Node-red** que podemos comenzar con una nueva iteración en la cadena para generar datos.

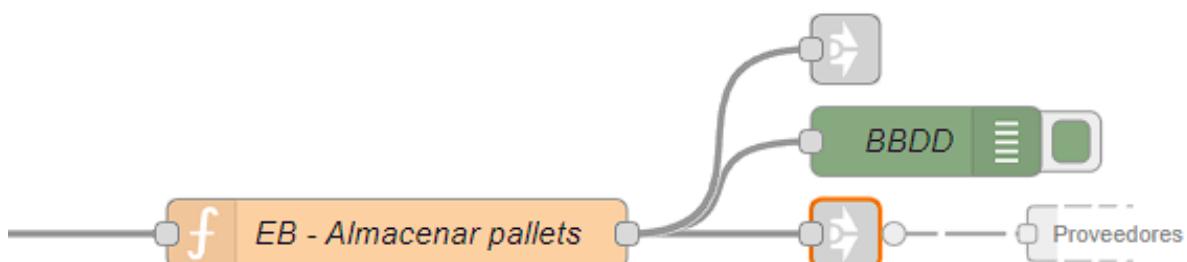


Figura 5.21: Node red - Proceso EB

Con todos los requisitos de información cumplidos, es el momento de empezar a generar los datos. El tiempo medio para recorrer una cadena completa desde que sale del proveedor hasta ser almacenado el productos es de 1 minuto. Dejando funcionar el Flujo **Node-red** que hemos configurado, hemos generado y guardado en **MongoDB** una cantidad de 162.235 documentos, lo que simula un total de 1.451 cadenas completas.

Como hemos podido observar, no todos los datos serán utilizados para hacer predicciones, en algunos casos solo mostraran información del proceso que se ha realizado. Por lo que podríamos hacer una división de los datos en tres categorías; correctos, erróneos y de información. Un dato sera determinado como correcto o

erróneo dependiendo si cumple o no las restricciones cuantitativas impuestas en cada proceso y de información si el proceso no requiere de restricciones.

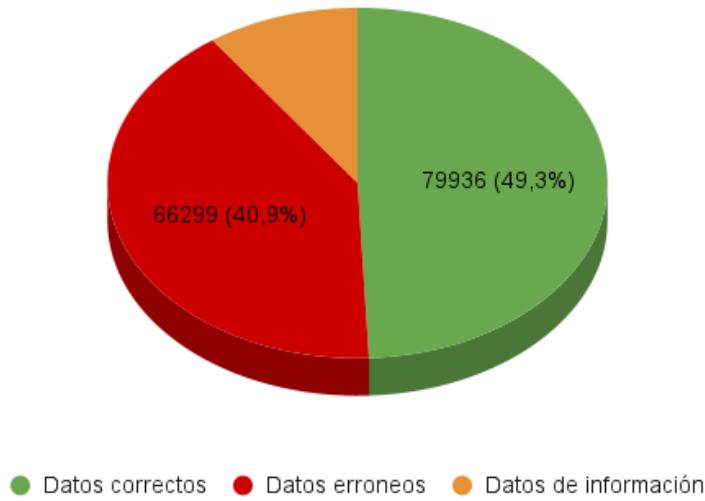


Figura 5.22: Datos guardados en MongoDB

#### 5.4.2 Procesamiento de datos

Esta sección será la encargada de analizar y procesar todos los datos generados en **Node-red**. Esto lo haremos con **Apache Spark**, una herramienta muy potente cuando nuestro fin es este mismo, analizar y tratar un gran volumen de datos. Empezaremos creando un proyecto en el entorno de desarrollo **Pycharm**, muy recomendado si estamos desarrollando proyectos **Python** al ofrecer funcionalidades como inspeccionador de código, indicador de errores sobre la marcha y refactorización de código automática. Añadimos las dependencias para **Hadoop** en su versión 2.7 y **Spark** en la versión 3.0.3 y las variables de entorno HADOOP\_HOME y SPARK\_HOME si estamos desarrollando en Windows.

La clase principal **spark\_submit** se encargara de dar funcionalidad a nuestra aplicación Spark, como veremos a continuación tenemos varias acciones disponibles dentro de esta clase y sera el controlador back-end quien especifique el parámetro concreto para realizar dicha acción.

---

```
python spark_submit.py PARAMETRO
```

---

#### Preparar los datos

Usaremos el parámetro **prepare** desde el script spark\_submit.py cuando queramos transformar los datos recogidos por los sensores en valores de entrada validos

para el modelo de aprendizaje automático, este paso previo resulta indispensable para el posterior entrenamiento del modelo. En el leeremos todos los documentos de las colecciones TFGCollectionCorrecto y TFGCollectionFallosTarea, la primera colección guarda todos los procesos que han cumplido con los requisitos de calidad y que han completado la cadena sin problemas, la segunda colección contiene aquellos procesos que por el contrario han sufrido fallo. Se han dividido en dos colecciones distintas para ayudarnos en el proceso de entrenamiento ya que el modelo requiere una columna indicando la categoría a la que pertenece, en nuestro caso es una distribución binaria de categorías, cero o uno dependiendo si el proceso ha tenido fallos o no.

MongoDB nos ofrece un conector con Spark que materializa los documentos leídos de MongoDB como DataFrames, aunque Spark también nos ofrece conversión a otros formatos como JSON o CSV. Comenzamos presentando el objeto `SparkSession`, introducido en la versión 2 de Spark, que nos permite acceder a las funcionalidades de esta herramienta y llamar a las API relacionadas con `DataFrame` y `Dataset`.

La inicialización de el objeto `SparkSession` se hace con las funciones **`builder`** y **`getOrCreate`** para crear la `SparkSession` si aún no existe o reutilizar una sesión existente. Con la función **`master`** podemos especificarle el cluster al que pertenece la aplicación Spark o determinar si ejecutamos Spark de forma independiente como es nuestro caso, al configurarlo como "`local[*]`" utilizara todas las CPU disponibles en el equipo. Para agregar alguna configuración concreta podemos usar la función **`config`** que recibe dos parámetros, uno con el nombre de la propiedad a modificar y otro con el valor de dicha propiedad. Una de estas configuraciones es la que nos permite conectar MongoDB y Spark aunque Como nuestro proyecto requiere de acceder a distintas colecciones, ha sido mas conveniente pasar estos valores por parámetros de entrada a la función **`get_spark_session`**.

---

```
def get_spark_session(db_in, db_out):
    spark_session = SparkSession
        .builder
        .appName("pyspark_getting_started")
        .master("local[*"])\ \
        .config("spark.mongodb.input.uri", db_in)
        .config("spark.mongodb.output.uri", db_out)
        .getOrCreate()
    return spark_session
```

---

Con una sesión de **Spark** ya operativa, si podemos hacer uso del conector que nos proporciona la librería de **Spark** para extraer los documentos de la base de datos y guardarlos localmente, esta nos permite especificarle algunas funciones si fuera necesario, como **`coalesce(number)`** que permite agrupar los documentos

leídos en la cantidad de grupos que especifiquemos.

```
spark_session.read.format("mongo").load().coalesce(1).write.option("header",
    "true").json(os.path.join(config.ROOT_DIR, path), mode="overwrite")
```

Los datos que utilizaremos para entrenar el modelo de aprendizaje automático, tiene una estructura distinta a los datos en bruto que guardamos en la base de datos. Esencialmente, estos siguen dos patrones dependiendo de las etapa en la que se hayan generado.

Transporte B y D	Fabricacion CB, CC, CD y CE
-- timestamp: long	-- timestamp: long
-- stage: string	-- stage: string
-- task: string	-- task: string
-- worker: string	-- id: string
-- order: string	-- trazabilityId: string
-- trazabilityId: string	-- temperature: double
-- temperature: double	-- preasure: double
-- humidity: double	
-- latitude: double	
-- longitude: double	

El objetivo es unificar y normalizar los datos recogidos en el transcurso de cualquiera de las etapas en datos para que puedan ser consumidos por el modelo. El modelo de aprendizaje automático utilizado en este proyecto, requiere la siguiente estructura.

Spark ML Model	
-- task: string	
-- milktank: string	
-- minTemperature: double	
-- maxTemperature: double	
-- avgTemperature: double	
-- minPreasure: double	
-- maxPreasure: double	
-- avgPreasure: double	
-- minHumidity: double	
-- maxHumidity: double	
-- avgHumidity: double	
-- goodness: integer	

La principal diferencia entre ambas estructuras de datos es la forma de agruparlos. En la etapa de transporte cada camión tiene un numero de lote al que pertenece cuatro tanques de leche y sera necesario obtener los valores mínimo, máximo y medio de temperatura y humedad que ha tenido cada uno en todo el viaje, agregando

## 5.4. GENERACIÓN Y PROCESAMIENTO DE LOS DATOS

las columnas **minPleasure**, **maxPleasure** y **avgPleasure** como vacíos, ya que a pesar de que en esta etapa no se generen estos valores, es requerido como entrada en el modelo de Spark. Por ultimo si el documento ha sido extraído de la colección TFGCollectionCorrecto añadiremos la constante 1 a la columna **goodness** del Dataframe y 0 si proviene de la colección TFGCollectionFallosTarea, esto lo utilizaremos como identificador de la categoría que queremos predecir. Es por eso que como se muestra a continuación, todos los tanques de leche de un mismo pedido comparten los mismos valores.

task	milk tank	minTemperature	maxTemperature	avgTemperature	minPleasure	maxPleasure	avgPleasure	minHumidity	maxHumidity	avgHumidity	goodness
B QGR3XDMGJI	0.7568953711778226 4.814977372388677 2.560377407189412	null	null	null	60.01539395985132 79.30958999857695 70.98255799129046	1					
B HV03LGBZAA	0.7568953711778226 4.814977372388677 2.560377407189412	null	null	null	60.01539395985132 79.30958999857695 70.98255799129046	1					
B S93SKAVOOX	0.7568953711778226 4.814977372388677 2.560377407189412	null	null	null	60.01539395985132 79.30958999857695 70.98255799129046	1					
B 1NKZVPDCCO	0.7568953711778226 4.814977372388677 2.560377407189412	null	null	null	60.01539395985132 79.30958999857695 70.98255799129046	1					
B SJQG02LB7R	0.026520905311673282 4.8621859486777781 6513802578099441	null	null	null	61.3987863548934 78.66987888306949 69.91120042338721	1					
B 2F7BYW2IR0	0.026520905311673282 4.8621859486777781 6513802578099441	null	null	null	61.3987863548934 78.66987888306949 69.91120042338721	1					
B FL7BYW2IR0	0.026520905311673282 4.8621859486777781 6513802578099441	null	null	null	61.3987863548934 78.66987888306949 69.91120042338721	1					
B ZLXIWW09MO	0.026520905311673282 4.8621859486777781 6513802578099441	null	null	null	61.3987863548934 78.66987888306949 69.91120042338721	1					
B B032WRJ3YX	0.09303106698501873 4.691150849988248 1.7790856561500037	null	null	null	60.10958030924433 78.23958857392124 68.1640406010455	1					
B ORNWLJN4GP	0.09303106698501873 4.691150849988248 1.7790856561500037	null	null	null	60.10958030924433 78.23958857392124 68.1640406010455	1					

Figura 5.23: Datos guardados en MongoDB

En cambio en la etapa de fabricación los tanques de leche son procesador individualmente, por esta razón y como se muestra a continuación, en cada tanque de leche se recogen valores distintos. Como sucedería en un caso practico real, si los productos que son transportados sufren alguna complicación, como un exceso de temperatura, son todos los productos los afectados independientemente de tanque de leche al que pertenezca dentro del vehículo. Esto no pasa la etapa de fabricación donde si sucede algún tipo de error en alguno de sus procesos, este solo afectara al tanque de leche donde se produjo ese error.

Para transformar la estructura de datos de esta etapa al modelo de entrenamiento, obtenemos los valores mínimos, máximos y medios de temperatura y presión en cada proceso. Agregamos las columnas vacias **minHumidity**, **maxHumidity** y **avgHumidity** que son necesarias para el modelo y establecemos una categoría modificando el valor de la columna **goodness** del Dataframe, de la misma manera que la etapa de transporte.

task	milk tank	minTemperature	maxTemperature	avgTemperature	minPleasure	maxPleasure	avgPleasure	minHumidity	maxHumidity	avgHumidity	goodness
CB F2385BZAI	-1.4613507254477947 7.441957616335269 4.037719449566221 0.3433634323744935 2.485568115880645 1.313989575145468	null	null	null	1	0					
CB FXBGET1GB	-3.2930990388963073 5.771776426331228 0.967573021365304 0.19275821774820567 1.2971496104074989 0.928038683981475	null	null	null	1	0					
CB 3Z320RM89F	-1.4044095204653169 4.375915648838807 0.9384834955452916 0.8330751428148804 2.678491719236077 1.6340753752888908	null	null	null	1	0					
CB OUZCQFOAZM	-2.2945378788788275 4.32865252697176 0.2596980807177567 0.32624532476517176 1.083913485145278 0.7325684524702423	null	null	null	1	0					
CB ROWKWE74ZM	-1.4006782343261532 6.994565552390887 2.524101602500745 0.3195759642744004 2.1222732283137598 1.210015509235986	null	null	null	1	0					
CB 90KFXV3NG0	1.8362439376709254 7.260080358540587 1.18542834607012 2.822000426098467 1.8061656026844175	null	null	null	1	0					
CB V6BMQEUVQ	2.5026827555788422 6.752983711737942 4.935773998494923 0.3091133887568967 2.84882754894329 1.102163952594445	null	null	null	1	0					
CB Z7VJP824MV	-2.0767171253474035 7.219203184784422 2.097030817124997 0.37901492399407943 1.440715414470898 0.9077307188768327	null	null	null	1	0					
CB WAD1JC1J18	-1.9869261360700148 5.384431940144875 1.84151221745826 0.808788506299773 1.3311113767245846 1.0305671530261635	null	null	null	1	0					
CB 9JUCVGU17A	3.267114250422058 7.05074135607098 4.754804856656643 0.23428667237148454 2.789563839097544 1.4671090781526193	null	null	null	1	0					

Figura 5.24: Datos guardados en MongoDB

El siguiente código muestra como hemos implementado en Spark estas transformación. Con la función **groupBy** recopilamos los datos iguales en grupos de Dataframse para realizar funciones agregadas en los datos agrupados. Las funciones

agregadas son tareas aritméticas que actúan en varias filas a la vez y devuelven un único valor agregado como resultado. En nuestro caso como requerimos del calculo de varios agregados, usamos la función **agg** dando como parámetros las distintas funciones Spark utilizadas con el objeto **f** y el nombre del campo que estemos calculando. Con la función **alias** podemos renombrar la columna del Dataframe a la que estamos realizando el agregado. Por ultimo como estamos mostrando un ejemplo de la etapa de fabricación, filtramos los procesos que pertenezcan a esta etapa y añadimos las columnas necesarias con la función **withColumn**.

---

```
grouped_by_milktank_factory = df.groupBy("task", "milktank").agg(
    f.min("temperature").alias("minTemperature"),
    f.max("temperature").alias("maxTemperature"),
    f.avg("temperature").alias("avgTemperature"),
    f.min("preasure").alias("minPreasure"),
    f.max("preasure").alias("maxPreasure"),
    f.avg("preasure").alias("avgPreasure"))
    .filter((df.task == "CB") | (df.task == "CC") |
    (df.task == "CD") | (df.task == "CE"))
    .withColumn("minHumidity", lit(None))
    .withColumn("maxHumidity", lit(None))
    .withColumn("avgHumidity", lit(None))
```

---

Por ultimo guardamos el Dataframe obtenido para ser utilizado en el entrenamiento del modelo .

### Entrenar el modelo

Con el parámetro **train** indicamos a la aplicación Spark que la acción que queremos realizar es el entrenamiento de los modelos de aprendizaje automático. Comenzamos por leer los datos transformados por el comando **prepare**. Estos son divididos en cinco Dataframes, uno por proceso en la etapa de faccionario (CB, CC, CD y CE) y otro agrupando las dos etapas de transporte (B y D) con el objetivo de generar un modelo por cada proceso y poder realizar predicciones justo cuando la tarea haya finalizado y decidir si dicho producto debe seguir tratándose o no. Para implementar esta funcionalidad y no repetir mucho código, se ha creado la clase **MultilayerPerceptronClassifier\_for\_iot** que por medio de la función **apply** podremos concretar que parámetros vamos a modificar. Recibirá dos parámetros, el primero es el Dataframe va a utilizar como conjunto de datos de entrenamiento, el segundo el nombre del proceso que vamos a entrenar.

---

```
MultilayerPerceptronClassifier_for_iot.apply(df_transport, "B_D")
MultilayerPerceptronClassifier_for_iot.apply(df_factory_CB, "CB")
```

---

La librería **Spark MLLib** nos permite implementar y entrenar modelos de Machine Learning de múltiples algoritmos de clasificación, regresión, clustering y fil-

trado colaborativo. En nuestro caso hemos optado por implementar un clasificador de perceptrones multicapa (MLPC). MLPC esta formado por varias capas de nodos. Cada capa está completamente conectada a la siguiente capa en la red. Los nodos en la capa de entrada representan los datos de entrada al modelo y que nosotros hemos denotado como **inputCols**. Todos los demás nodos asignan entradas a salidas mediante una combinación lineal de las entradas con los pesos y sesgos del nodo y aplicando una función de activación. Los nodos en las capas intermedias usan la función sigmoidea (logística) y los nodos en la capa de salida usan la función softmax. La función de activación sigmoidea es un tipo de función de activación logística, se utiliza en las capas ocultas de las redes neuronales para transformar la salida lineal en una no lineal. La función de activación de Softmax se utiliza en la capa de salida de las redes neuronales para convertir la salida lineal en probabilista.

Comenzamos dividiendo el conjunto de datos del dataframe en dos grupos, uno contendrá el 90 por cierto del dataframe y lo usaremos para entrenar al modelo, el otro 10 por cierto restante es usado para verificar y comprobar la eficacia del algoritmo. Ademas guardamos en una lista el nombre de las columnas que el modelo tomara como entrada, para este caso descartaremos los valores "task" y "milktank" y conservamos los demás.

---

```
trainDF, testDF = df.randomSplit([.9, .1], seed=43)
inputCols = trainDF.columns[2: len(trainDF.columns)]
```

---

El siguiente fragmento de código es la implementación de dicho modelo en pyspark, con tres capas ocultas de 72, 144 y 72 neuronas respectivamente.

---

```
layers = [len(trainDF.columns) - 2, 72, 144, 72, 2]
mlp = MultilayerPerceptronClassifier(labelCol=labelCol,
                                       maxIter=100, layers=layers, blockSize=128, seed=1234)
```

---

Los pipelines definen las fases y el orden de un proceso de aprendizaje automático. En MLLib, las fases de una canalización o transferencia se representan mediante una secuencia específica, donde un transformador y un estimador realizan tareas. Un estimador es una abstracción de algoritmos de aprendizaje y es responsable del ajuste o el aprendizaje en un conjunto de datos, el estimador implementa un método denominado **fit()** para producir un elemento transformador.

---

```
pipeline\_\_mlp = Pipeline(stages=[mlp])
pipelineModel\_\_mlp = pipeline\_\_mlp.fit(trainDF)
```

---

Para poder hacer predicciones sin tener que entrenar el modelo cada vez, sera conveniente guardar este con la siguiente nomenclatura "**model\_** + task" donde **task** es el nombre del proceso que estamos entrenando.

---

```
modelPath = os.path.join(config.ROOT_DIR,
    'data/models/model_' + task + '/multilayerPerceptronClassifier')
pipelineModel_mlp.write().overwrite().save(modelPath)
```

---

Esta es una instancia de un Evaluador. Permite calcular alguna medida común para tareas de clasificación como la precisión.

---

```
prediction_dataframe_mlp = pipelineModel_mlp.transform(testDF)
evaluator = MulticlassClassificationEvaluator(labelCol=labelCol,
    predictionCol="prediction", metricName="accuracy")
mlp_accuracy = evaluator.evaluate(prediction_dataframe_mlp)
```

---

Los resultados obtenidos muestran cuantos elementos había para cada proceso y cual ha sido su precisión y error, en nuestro caso hemos obtenido una precisión del 100 %.

B_D : 10452	CD : 5224
Accuracy of MLP Classifier is = 1	Accuracy of MLP Classifier is = 1
Error of MLP Classifier is = 0	Error of MLP Classifier is = 0
CB : 5224	CE : 5224
Accuracy of MLP Classifier is = 1	Accuracy of MLP Classifier is = 1
Error of MLP Classifier is = 0	Error of MLP Classifier is = 0
CC : 5224	
Accuracy of MLP Classifier is = 1	
Error of MLP Classifier is = 0	

---

## Hacer predicciones

La ultima funcionalidad desarrollada para la aplicación Spark se obtiene con el parámetro **predict**, con este leeremos los datos que se encuentran en las colección TFGCollectionToProcess y realizaremos predicciones para almacenar su resultado en TFGCollectionProcessed. Cargamos también el modelo del Pipeline que guardamos con el parámetro **train**. Los modelos de tipo Pipeline nos dan la posibilidad implementar distintos estimadores y transformadores para ejecutarlos secuencialmente y crear modelos mas complejos.

---

```
pipelineModel_mpc_B_D = PipelineModel.load(
    os.path.join(config.ROOT_DIR, 'data/models/model_B_D/
        multilayerPerceptronClassifier'))
```

---

Un transformador es un componente de alto nivel que puede agregar, eliminar o actualizar las características existentes en el marco de datos. Cada transformador tiene un método **transform()** que se llama cuando se ejecutan el pipeline.

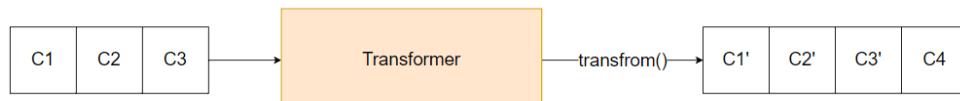


Figura 5.25: Estructura Transformer

Un estimador es un componente de aprendizaje, el modelo devuelto transforma el marco de datos de acuerdo con los parámetros que se ajustan durante la fase de aprendizaje. Cada estimador tiene un método **fit()** que devuelve un Modelo que a su vez tiene un método **transform()**.

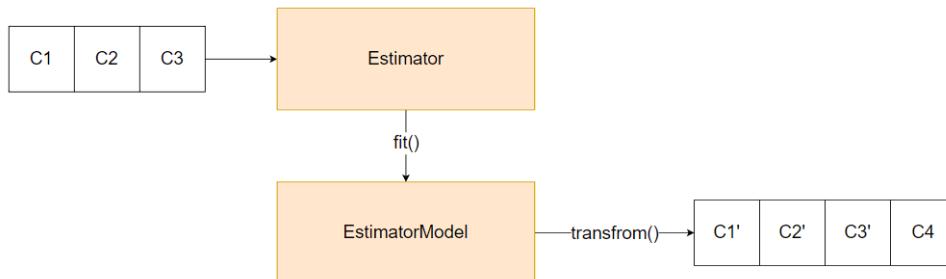


Figura 5.26: Estructura Estimator

Para hacer predicciones con el modelo ya cargado, filtramos aquellos procesos que vayamos a predecir. Si por ejemplo como se muestra en el fragmento de código siguiente, queremos hacer predicciones sobre la etapa de transporte, el modelo que necesitamos para eso es pipelineModel\_mpc\_B\_D ya que fue entrenado con datos de transporte, en el código de nuestra aplicación Spark se cargan todos los modelos y se hace predicciones de todas los procesos. También añadimos la columna "stage" que estableceremos con el valor "Prediction" al marco de datos para cuando lo almacenemos en **MongoDB** tenga una etiqueta acorde con la función realizada.

---

```

df_transports_predictions = pipelineModel_mpc_B_D.transform(df_input_model
    .filter((df_input_model.task == "B")|(df_input_model.task == "D"))
    .select(config.transport_fields))
    .drop("goodness")
    .withColumn("stage", lit("Prediction"))
  
```

---

Usando de nuevo el conector de MongoDB con Spark, guardamos el conjuntos de datos procesados y las predicciones hechas en la colección TFGCollectionProcessed.

---

```

df_transports_predictions.drop("features", "scaledFeatures",
    "rawPrediction", "probability")
    .write.format("mongo")
    .option('uri', config.mongo_processed_uri + config.
        sufix_secuity_mongo_config)
  
```

---

---

```
.option('database', 'TFG')
.option('collection', 'TFGCollectionProcessed')
.mode("append").save()
```

---

Las predicciones guardadas en MongoDB tienen la siguiente estructura.

```
{
  "stage": "Prediction",
  "task": "CB",
  "milktank": "TCQABMTPOS",
  "minTemperature": 2.411531616228684,
  "maxTemperature": 3.6674252583360287,
  "avgTemperature": 3.121600923727608,
  "minPleasure": 0.7938937050280519,
  "maxPleasure": 1.2299624274803984,
  "avgPleasure": 0.9750132408659378,
  "prediction": 1
}
```

---

Por ultimo y para no tener redundancia de información en la base de datos, borramos todos los documentos guardados en TFGCollectionToProcess que ya han sido procesados.

---

```
mongo_client = MongoClient(config.mongo_uri)
collectionToProcess = mongo_client[config.database][config.
    collectionToProcess]
collectionToProcess.delete_many({})
```

---

## 5.5 Back-end

El back-end se encarga de dar funcionalidad al proyecto, gestionando las llamadas a la aplicación y comunicándose con la base de datos. Esta escrita en Java con el framework **Spring Boot** que nos facilita el desarrollo y nos automatiza tareas como la inyección de dependencias o el despliegue de la aplicación. Java desde su versión 5 incluyó soporte para las **anotaciones** y la mayoría de framework incluido Spring adoptaron las grandes posibilidades que estas añaden al framework. Una arquitectura de este tipo nos permite crear una aplicación más compleja por la unión de aplicaciones más pequeñas. Estas pueden ser escaladas, manejadas y desarrolladas de forma independiente, ya que pueden estar escritas en distintos lenguajes. Al tener la libertad de elegir la tecnología más idónea para cada parte del proyecto, podemos aprovechar al máximo las distintas funciones y características de esta, además de poder escalar y abaratar el proyecto más eficientemente.

Las anotaciones que hemos utilizado con mas regularidad son las siguientes, aunque cuando expliquemos los distintos modelos disponibles, añadiremos algunas mas.

La anotación @RestController se aplica a una clase para identificarla como controlador de llamadas y se usa para crear servicios web RESTful usando Spring MVC. Spring RestController se encarga de asignar los datos de las llamadas al método del controlador definido. Una vez que el cuerpo de la respuesta se genera a partir del método del controlador, lo convierte en una respuesta JSON. Las API son conjuntos de reglas que son utilizados para diseñar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde el consumidor establece el contenido que se necesita por una la llamada y el productor solicita una respuesta.

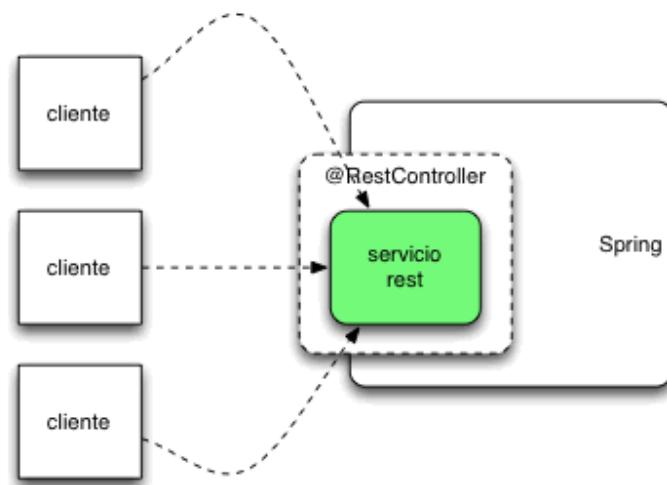


Figura 5.27: Rest Service [Internet]. Arquitecturajava.com. 2022 [cited 2022 Nov 27]. Available from: <https://www.arquitecturajava.com/wp-content/uploads/springrestservicepublicacion1.gif>

Para las siguientes anotación es necesario comprender que un Java Bean es un estándar que cumple ciertas normas con el nombre de sus propiedades y métodos, que nos permite establecer características específicas a una clase Java. Es por tanto un componente que nos permite encapsular su contenido para ofrecerle una mejor estructura y poder reutilizar el código en otras clases. Los Java Beans deben de cumplir ciertas restricciones como un constructor sin argumentos, atributos privados con todos los getters y setters de cada uno.

@Autowired es una anotación que permite inyectar dependencias en la clase requerida. Al declarar todas las dependencias de beans en el proyecto Spring Boot, el contenedor de Spring puede conectar automáticamente las relaciones entre los

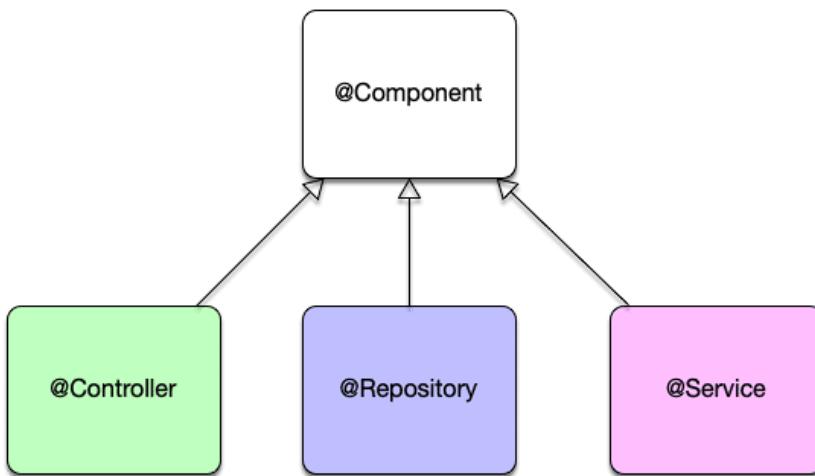


Figura 5.28: Herencia de anotaciones

beans que colaboran. Esto se llama cableado automático de Spring Bean.

Colocando la anotación `@SpringBootApplication` en la cabecera de la clase principal de la aplicación Spring, escaneará automáticamente los componentes del paquete actual y sus subpaquetes. Esto se registrará en el contexto de la aplicación y nos permitirá inyectar dependencias con la anotación antes mencionada, `@Autowired`.

Otra anotación que se incluye en la cabecera de la aplicación Spring es `@EnableMongoRepositories` que nos permite activar el uso de repositorios MongoDB y que hemos completado de configurar en el archivo **application.properties** mediante las variables:

---

```

spring.data.mongodb.uri=mongodb+srv://user:pass@cluster0.s76am.mongodb.net
spring.data.mongodb.database=TFG
  
```

---

La anotación `@Component` nos sirve para clasificar una clase de forma genérica, marca la clase Java como un bean o componente para que el mecanismo de exploración de componentes de Spring pueda agregarla al contexto de la aplicación. Nos es usual utilizarla, lo esperado es usar alguna de sus anotaciones derivadas como son `@Repository`, `@Service` y `@Controller` para las capas de persistencia, servicios y presentación respectivamente.

`@Service` caracteriza una clase Java que realiza algún servicio, como ejecutar lógica de negocios, realizar cálculos y llamar a API externas.

---

```

@Service
public class SparkService {
  
```

---

`@Repository` se usa para indicar que la clase proporciona mecanismos para las

operación de almacenamiento, recuperación, búsqueda, actualización y eliminación de objetos propias de una base de datos. Dentro de esta clase podemos usar la anotación `@Query` para declarar consultas de búsqueda directamente por método.

---

```
@Repository
public interface GenericTaskRepository extends MongoRepository<GenericTask,
    String> {

    @Query("{task:'?0'}")
    List<GenericTask> findProcessByTask(String task);
```

---

`@Controller` sirve para señalar los controladores que son los encargados de enlazar con la capa de presentación y de registrar las diferentes URLs a las que el componente responde a través de los diferentes `@RequestMappings` o `@GetMapping`.

los modelos utilizados:

**ProcessedTask** este modelo corresponde con el objeto que guardamos en la base de datos cuando realizamos una predicción sobre un tanque de leche.

**GenericTask** es un modelo que tiene todos los atributos de los procesos.

**StorageModel** es el modelo de almacenamiento que se utiliza para identificar el pallet dentro del almacén.

**UserAuthModel** este modelo se utiliza en el inicio de sesión de la aplicación.



Figura 5.29: Modelos utilizados en Java

Todas las clases que definen un modelo, están encabezadas con las anotacio-

nes @Data y @Document. @Data genera los getter y setters de todos los campos de la clase, ademas podemos sobrescribir métodos por defecto con la anotación @Override como solemos hacer con el método **toString()**. Con la anotación @Document(collection="collectionName") indicamos la colección donde sera almacenado el objeto de la clase a la que pertenezca. Dentro de la clase podemos mencionar dos anotaciones, @Id indica cual de los atributos representa al identificador del objeto y @Field establece los metadatos personalizados para el documento en MongoDB.

---

```

@Data
@Document(collection = "TFGCollectionProcessed")
public class ProcessTask {

    @Id
    private String id;

    @Field("stage")
    private String stage;

    @Field("task")
    private String task;

```

---

la lista de servicios que las aplicación ofrece son.

loginAuth(String user, String password) este método se usa para identificar al usuario al iniciar sesión en la aplicación. Para realizar esta lógica, el método recibe los parámetros user y password y realizara una consulta en la colección Users de MongoDB con los parámetros mencionados. Si existe un documento para esos valores, el método devolverá el mensaje "LOGIN:SUCCESS" y "LOGIN:REJECT" en caso contrario.

sparkAction(String action) Este método permite ejecutar la aplicación Spark que explicamos en la sección anterior con alguno de las funciones prepare, train o predict. Cuando la ejecución termina, el método recibe una respuesta en forma de log indicando la acción realizada, el momento y el tiempo que ha requerido para la ejecución. Esta respuesta no esta siendo procesada pero podría ser guardada o mostrada al usuario para tener un mejor control del estado de los datos.

---

```
{ action: train, timestamp: 12345678 , processingTime: 73 }
```

---

getProcesses(String trazabilityId) este método realizara una consulta con los procesos involucrados en un determinado identificado de trazabilidad (trazabilityId) y devolverá todos los documentos con este identificador como un conjunto de GenericTask que usara la vista para representar los resultados.

getPredictedProcesses(String milkTank) y getMultiplePredictedProcesses(List<String> ■

milkTanks) realizan una consulta a MongoDB para obtener las predicciones de los tanque de leche que se le especifique por parámetros, el método devolverá un conjunto de ProcessedTask.

## 5.6 Aplicación multiplataforma

La ultima sección sera la encargada de mostrar los datos que hemos ido reco-  
giendo y tratando a lo largo de la cadena, al crear una aplicación para esto nos  
permite visualizar, incluso controlar el estado de la empresa en tiempo real. La  
aplicación esta escrita en **TypeScript**, una envoltura de **JavaScript** y desarrollada  
con el Framework Ionic en su version 6. Ionic nos permite desarrollar aplicacio-  
nes híbridas basadas en las tecnologías web HTML, CSS y JS. En contra posición  
las aplicaciones nativas se desarrollan en el lenguaje nativo del dispositivo. Depen-  
diendo de la plataforma tendremos que utilizar un lenguaje de programación y un  
entorno de desarrollo específico y tener un mayor conocimiento de las propiedades  
nativas del sistema operativo de destino. Una consecuencia negativa cuando que-  
remos encontrar un perfil debido a las grandes diferencias entre IOS y Android.  
Un rasgo positivo de las aplicaciones nativas es que ofrecen un mayor rendimiento  
que las híbridas ya que están escritas y optimizadas en el lenguaje propio del dis-  
positivo. En nuestro caso, el rendimiento no es un factor critico y las aplicaciones  
híbridas como la que podemos desarrollar con Ionic nos ofrece algunas funciona-  
lidades para allanarnos el camino cuando estamos desarrollando ademas de poder  
ser desplegada en IOS o Android con el mismo código.

En la figura §5.30 tenemos una primera visualización de la aplicación termina-  
da. La captura de pantalla de la izquierda corresponde con el inicio de sesión y la  
captura de la derechera con la primera vista al acceder a la trazabilidad de produc-  
tos. El inicio de sesión usa el método loginAuth(String user, String password) y si  
el servicio devuelve "LOGIN:SUCCESS" se le permitirá el acceso a la trazabilidad  
de productos.

Cuando accedemos a la aplicación, tenemos una vista principal con cinco iconos  
circulares, uno por cada etapa en la cadena. Cada etapa nos muestra los procesos  
que intervienen en cada una, los procesos son representados con estilos distintos  
en función del tipo de tarea que realiza. En la figura §5.31 tenemos todas las vistas  
disponibles para los procesos de trazabilidad, divididas en tres capturas, la captura  
de la izquierda representa en un gráfico los valores de temperatura y presión reco-  
gidos durante la tarea **CB**. Ademas, en este mismo proceso podemos ver como cada  
tanque de leche recoge valores distintos por ser procesados individualmente. En la  
captura central por el contrario todos los taques comparten la misma temperatura  
en cada proceso de transporte, ademas como podemos observar la aplicación Spark  
determinó que este proceso no cumple con los requisitos de calidad y lo marcó con

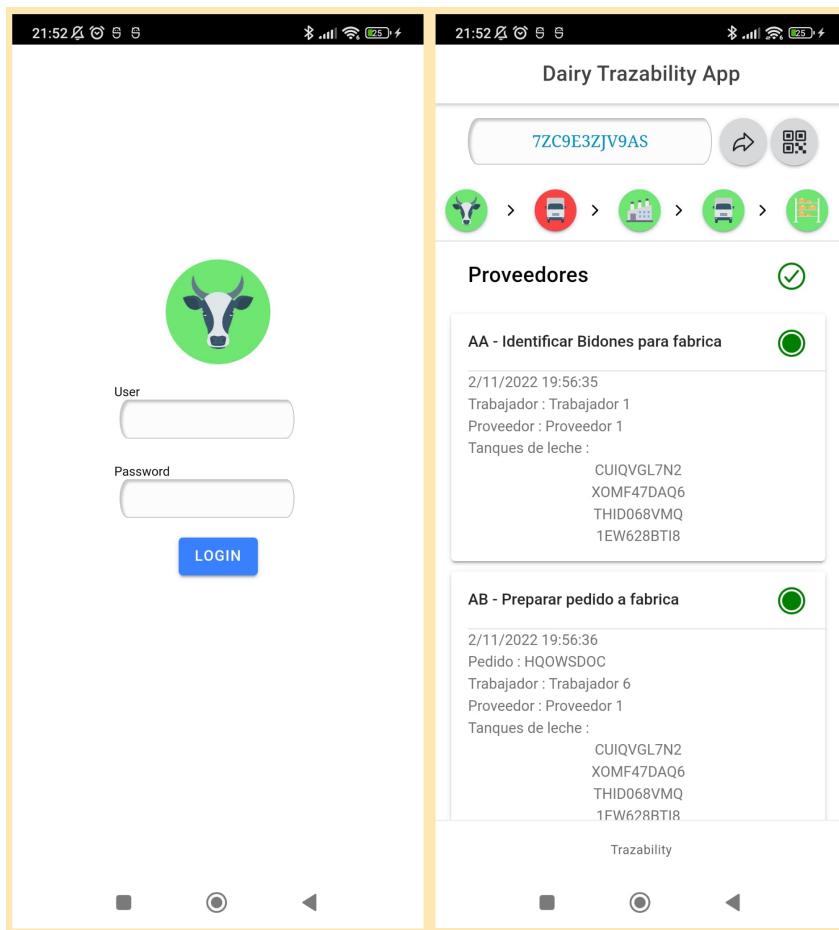


Figura 5.30: Login y vista principal

un contorno rojo, lo que tiene sentido ya que los valores recogidos para este caso son en su mayoría negativos. La ultima captura de la figura muestra el ruta que ha seguido el camión en el transporte 1.

Por ultimo tenemos en la parte superior, una entrada de texto con dos botones, uno consultara al back-end con el **trazabilityId** introducido. El otro botón sirve para escanear un código de barras y realizara la misma función, este es generado a partir del **trazabilityId**. En un entorno real este código de barras podría estan en un pallet o bidón pegado y solo es necesario escanearlo con la cámara del móvil para obtener su historial

## 5.6. APLICACIÓN MULTIPLATAFORMA

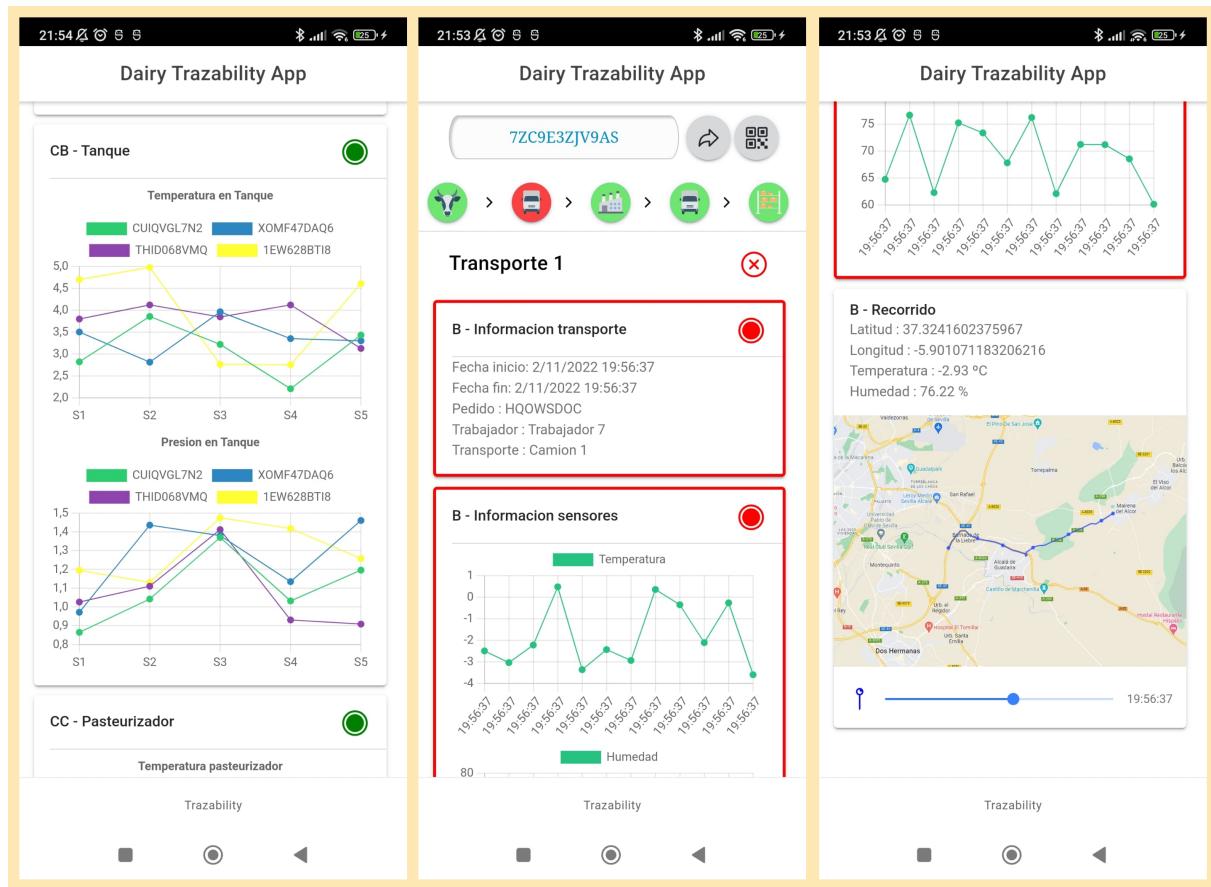


Figura 5.31: Vista de la aplicación

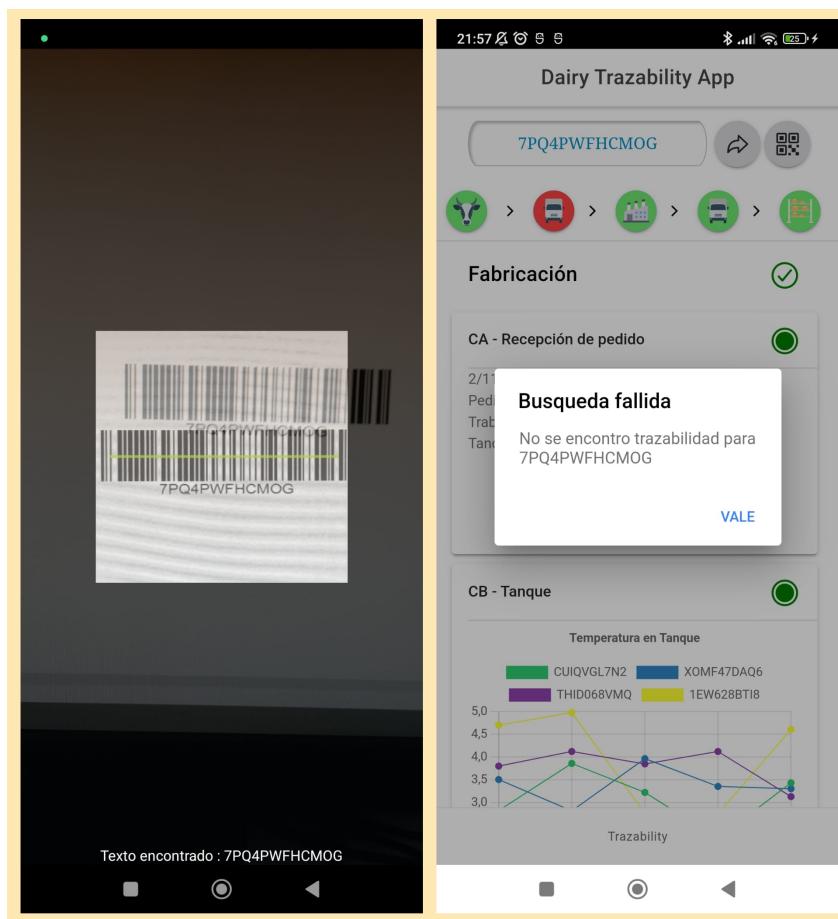


Figura 5.32: Vista de la aplicación

---

## **Parte IV**

---

### **Pruebas**

---

## Pruebas

**E**sta sección muestran las pruebas realizadas.

## 6.1 Prueba 1: Dispositivo IoT

Una visualización del dispositivo IoT finalizado



Figura 6.1: Prototipo dispositivo IoT

## 6.2 Prueba 2: Traza completa generada por un pedido

Objetos guardados en la base de datos y generado por cada uno de los procesos

AA - Identificar bidones

---

```
{
  "trazabilityId": "7ZC9E3ZJV9AS",
  "stage": "Proveedor",
  "task": "AA",
  "timestamp": 1667415395950,
  "worker": "Trabajador 1",
  "supplier": "Proveedor 1",
  "milkTanks": [
    "CUIQVGL7N2",
    "XOMF47DAQ6",
    "THID068VMQ",
    "1EW628BTI8"
  ]
}
```

---

## 6.2. PRUEBA 2: TRAZA COMPLETA GENERADA POR UN PEDIDO

### AB - Preparar pedido

```
{    "trazabilityId": "7ZC9E3ZJV9AS",
    "stage": "Proveedor",
    "task": "AB",
    "timestamp": 1667415396126,
    "worker": "Trabajador 6",
    "supplier": "Proveedor 1",
    "order": "HQ0WSDOC",
    "milkTanks": [
        "CUIQVGL7N2",
        "XOMF47DAQ6",
        "THID068VMQ",
        "1EW628BTI8"]
}
```

### B - Transporte 1

```
{    "timestamp": 1667415397971,
    "stage": "Transporte 1",
    "task": "B",
    "worker": "Trabajador 7",
    "transport": "Camion 1",
    "order": "HQ0WSDOC",
    "trazabilityId": "7ZC9E3ZJV9AS",
    "temperature": -2.493073885629024,
    "humidity": 64.7598622543746,
    "latitude": 37.367528231199266,
    "longitude": -5.767242623556242
}
```

### CA - Identificar tanques de leche

```
{    "stage": "Fabricacion",
    "task": "CA",
    "timestamp": 1667415398404,
    "worker": "Trabajador 11",
    "order": "HQ0WSDOC",
    "milkTanks": [
        "CUIQVGL7N2",
        "XOMF47DAQ6",
        "THID068VMQ",
        "1EW628BTI8"],
    "trazabilityId": "7ZC9E3ZJV9AS"
}
```

CB - Conservación en tanques

```
{  
    "stage": "Fabricacion",  
    "task": "CB",  
    "timestamp": 1667415398878,  
    "milkTank": "CUIQVGL7N2",  
    "temperature": 2.818322941290117,  
    "pressure": 0.8643708996665576,  
    "trazabilityId": "7ZC9E3ZJV9AS"  
}
```

CC - Proceso de pasteurización

```
{  
    "stage": "Fabricacion",  
    "task": "CC",  
    "timestamp": 1667415400923,  
    "milkTank": "CUIQVGL7N2",  
    "temperature": 79.69729210930927,  
    "pressure": 0.9951338722155484,  
    "trazabilityId": "7ZC9E3ZJV9AS"  
}
```

CD - Proceso de homogeneización

```
{  
    "stage": "Fabricacion",  
    "task": "CD",  
    "timestamp": 1667415418164,  
    "milkTank": "THID068VMQ",  
    "temperature": 68.18860282948802,  
    "pressure": 110.75578059926326,  
    "trazabilityId": "7ZC9E3ZJV9AS"  
}
```

CE - Proceso de embotellado

```
{  
    "stage": "Fabricacion",  
    "task": "CE",  
    "timestamp": 1667415413585,  
    "milkTank": "XOMF47DAQ6",  
    "temperature": 17.502371123739884,  
    "pressure": 1.171199209660584,  
    "trazabilityId": "7ZC9E3ZJV9AS"
```

## 6.2. PRUEBA 2: TRAZA COMPLETA GENERADA POR UN PEDIDO

}

---

### CF - Preparar pallets

```
{  
    "trazabilityId": "7ZC9E3ZJV9AS",  
    "stage": "Fabricacion",  
    "task": "CF",  
    "timestamp": 1667415430530,  
    "worker": "Trabajador 16",  
    "order": "22KFMWR0",  
    "pallets": [  
        "CUIQVGL7N2",  
        "XOMF47DAQ6",  
        "THID068VMQ",  
        "1EW628BTI8"  
    ]  
}
```

---

### D - Transporte 2

```
{  
    "trazabilityId": "7ZC9E3ZJV9AS",  
    "stage": "Transporte 2",  
    "task": "D",  
    "timestamp": 1667415430530,  
    "worker": "Trabajador 16",  
    "transport": "Camion 4",  
    "order": "22KFMWR0",  
    "temperature": 1.7818828642607032,  
    "humidity": 72.88985856992255,  
    "latitude": 37.329142459080686,  
    "longitude": -5.938369782611849  
}
```

---

### EA - Identificar pallets

```
{  
    "_id": {  
        "$oid": "6362bd894d83681154964524"  
    },  
    "stage": "Almacen",  
    "task": "EA",  
    "timestamp": 1667415433516,  
    "worker": "Trabajador 19",  
    "order": "22KFMWR0",  
    "pallet": "CUIQVGL7N2"  
}
```

```

    "pallets": [
        "CUIQVGL7N2",
        "XOMF47DAQ6",
        "THID068VMQ",
        "1EW628BTI8"
    ],
    "trazabilityId": "7ZC9E3ZJV9AS",
    "_msgid": "df726f27.de7ea",
    "_event": "node:2cac2744.8e4cd8"
}

```

---

## EB - Almacenar pallets

```

{
    "trazabilityId": "7ZC9E3ZJV9AS",
    "stage": "Almacen",
    "task": "EB",
    "timestamp": 1667415433718,
    "worker": "Trabajador 24",
    "order": "22KFMWR0",
    "storage": [
        {
            "pallet": "CUIQVGL7N2",
            "hall": "Pasillo 5",
            "number": 7
        },
        {
            "pallet": "XOMF47DAQ6",
            "hall": "Pasillo 3",
            "number": 10
        },
        {
            "pallet": "THID068VMQ",
            "hall": "Pasillo 1",
            "number": 8
        },
        {
            "pallet": "1EW628BTI8",
            "hall": "Pasillo 5",
            "number": 7
        }
    ]
}

```

---

### 6.3. PRUEBA 3: PREDICCIÓN SOBRE LA CALIDAD DEL PRODUCTO

#### 6.3 Prueba 3: Predicción sobre la calidad del producto

Como podemos ver en la imagen, en el procesos B (Transporte a la fabrica), la temperatura a la que se ha transportado la leche ha sido demasiado baja y el modelo ha pronosticado correctamente un error de calidad.

task	milk tank	minTemperature	maxTemperature	avgTemperature	minPressure	maxPressure	avgPressure	features	scaledFeatures	rawPrediction	probability/prediction	stage
CB  EWG28BT18  2, 74922270575120654   4.97515203854476   3.9570376935525   1.13026678575028   [1, 474868525246005   [1, 0, 2, 749222705750280739   [2, 009418226195490, ...   [-13, 026717954490, ...   [4, 1521076716654...   1.0 Prediction												
CB  CUIQVGL7N2  2, 20414315188244738   3.854763052808987   1, 0045617456130   0, 864370893857561   1, 36994630977441   1, 10796062380679   [1, 0, 2, 20414315188244738   [2, 009418226195490, ...   [-13, 026717954490, ...   [3, 75591403964959, ...   1.0 Prediction												
CE  XOMF47DA06  12, 811105747493117   3.9570376935525   1.13026678575028   [1, 474868525246005   [1, 0, 2, 749222705750280739   [2, 009418226195490, ...   [-13, 026717954490, ...   [1, 1521076716654...   1.0 Prediction												
CB  XOMF47DA06  12, 811105747493117   3.960485589359761   3.838870629857733   0, 07147738990218661   1, 4603161253215846   1, 276401158193884   [1, 0, 2, 811105747493117   [2, 009418226195490, ...   [-13, 490119947633, ...   [3, 64562838611211, ...   1.0 Prediction												
CC  CUIQVGL7N2  71, 126339219832447   79, 69729210930927   74, 5048798153115   0, 7644803001479019   1, 245644876109159   0, 923462689740538   [1, 0, 71, 326339219, ...   [2, 0087432277768, ...   [-10, 050681849813, ...   [7, 04094569279080, ...   1.0 Prediction												
CC  TH1D068W8M  72, 60391831625219   75, 672473979864338   73, 789169943598   0, 750027808206272   1, 1028762059984   0, 9469196031869448   [1, 0, 72, 603918316, ...   [2, 0087432277768, ...   [-11, 063820893213, ...   [2, 41070728650439, ...   1.0 Prediction												
CC  EWG28BT18  72, 20414315188244738   77, 74796518730845   75, 8385761857357   0, 750027808206272   1, 1028762059984   0, 9469196031869448   [1, 0, 72, 74796518730845, ...   [2, 0087432277768, ...   [-11, 063820893213, ...   [2, 41070728650439, ...   1.0 Prediction												
CC  XOMF47DA06  12, 2370612338456   3.9570376935525   1.13026678575028   [1, 474868525246005   [1, 0, 2, 749222705750280739   [2, 009418226195490, ...   [-13, 026717954490, ...   [1, 1521076716654...   1.0 Prediction												
CD  TH1D068W8M  60, 17944376752849   74, 14453976075913   66, 20088964470044   98, 72544135224622   147, 0608985606592   92, 128, 66570720483709   [1, 0, 60, 17944376752849, ...   [2, 00884465992114, ...   [-14, 466043340145, ...   [8, 38042475466695, ...   1.0 Prediction												
CD  CUIQVGL7N2  67, 54568659292738   79, 9667241371712   73, 89857567381712   99, 97730542390129   132, 61378514040453   114, 77055611042822   [1, 0, 67, 545686592, ...   [2, 00884465992114, ...   [-12, 960752643330, ...   [1, 61432818202116, ...   1.0 Prediction												
CD  EWG28BT18  69, 33747149548506   73, 3152320733415   71, 1051534516671   107, 74877689006118   138, 99263323547025   124, 5157858784894   [1, 0, 69, 337471495, ...   [2, 00884465992114, ...   [-13, 665751897076, ...   [4, 03629981173896, ...   1.0 Prediction												
CD  XOMF47DA06  60, 0950958659292738   78, 5385761857357   71, 89857567381712   115, 2079300780276   1129, 3443732540464   123, 454223410428   [1, 0, 60, 0950958659, ...   [2, 00884465992114, ...   [-13, 202105390280, ...   [1, 0339902833201, ...   1.0 Prediction												
CE  XOMF47DA06  12, 2370612338456   3.9570376935525   1.13026678575028   [1, 474868525246005   [1, 0, 2, 749222705750280739   [2, 009418226195490, ...   [-13, 026717954490, ...   [1, 1521076716654...   1.0 Prediction												
CE  CUIQVGL7N2  15, 7597019698076619, 010450621785335   17, 584633420930455   10, 9586094494883958   1, 176144349030095   1, 0750264691671416   [1, 0, 15, 759701969, ...   [2, 008893154311720, ...   [-11, 203759267979, ...   [8, 87289728390629, ...   1.0 Prediction												
CE  XOMF47DA06  15, 47301672039084   17, 59650952279712   16, 48787506189567   0, 933571828379538   1, 17119209660584   1, 00293450526066   1, 473016720, ...   [2, 0085935431720, ...   [-11, 473016720, ...   [6, 47383595339510, ...   1.0 Prediction												
CE  TH1D068W8M  15, 460914519606392   19, 947449195184838   18, 03076744955994   1, 00516235651342544   1, 23676419045717   1, 137548712042310   1, 0, 15, 460914519, ...   [2, 0085935431720, ...   [-10, 0438751260354, ...   [8, 3060197695406, ...   1.0 Prediction												
task	milk tank	minTemperature	maxTemperature	avgTemperature	minHumidity	maxHumidity	avgHumidity	features	scaledFeatures	rawPrediction	probability/prediction	stage
B  CUIQVGL7N2  -3, 595397249360943   0, 47479672828122   -1, 83037950163396   0, 144005947824986   16, 607642929539   [1, 0, 3, 595397249360943, ...   [2, 00900077048243, ...   [-16, 4403728639079, ...   [1, 99998002446860, ...   0.0 Prediction												
B  XOMF47DA06  -3, 595397249360943   0, 47479672828122   -1, 83137950163396   0, 144005947824986   16, 657642925539   [1, 0, 3, 595397249360943, ...   [2, 00900077048243, ...   [-16, 4403240639079, ...   [1, 99998002446860, ...   0.0 Prediction												
B  TH1D068W8M  -3, 595397249360943   0, 47479672828122   -1, 83137950163396   0, 144005947824986   16, 657642925539   [1, 0, 3, 595397249360943, ...   [2, 00900077048243, ...   [-16, 4403240639079, ...   [1, 99998002446860, ...   0.0 Prediction												
B  EWG28BT18  -3, 595397249360943   0, 475407672828122   -1, 83137950163396   0, 144005947824986   16, 657642925539   [1, 0, 3, 595397249360943, ...   [2, 00900077048243, ...   [-16, 4403240639079, ...   [1, 99998002446860, ...   0.0 Prediction												
D  CUIQVGL7N2  0, 0826266765952892   2, 47843355867732   1, 266981180780262   1, 292661867625778   96031042876001   68, 526738958583   [1, 0, 0, 0826266765, ...   [2, 00900077048243, ...   [-11, 50498605990, ...   [1, 3919483564267, ...   1.0 Prediction												
D  XOMF47DA06  0, 0826266765952892   2, 47843355867732   1, 266981180780262   1, 266981180780262   61, 292661867625778   96031042876001   68, 526738958583   [1, 0, 0, 0826266765, ...   [2, 00900077048243, ...   [-11, 50498605990, ...   [1, 3919483564267, ...   1.0 Prediction												
D  TH1D068W8M  0, 0826266765952892   2, 47843355867732   1, 266981180780262   1, 266981180780262   61, 292661867625778   96031042876001   68, 526738958583   [1, 0, 0, 0826266765, ...   [2, 00900077048243, ...   [-11, 50498605990, ...   [1, 3919483564267, ...   1.0 Prediction												
E  EWG28BT18  0, 0826266765952892   2, 47843355867732   1, 266981180780262   1, 266981180780262   61, 292661867625778   96031042876001   68, 526738958583   [1, 0, 0, 0826266765, ...   [2, 00900077048243, ...   [-11, 50498605990, ...   [1, 3919483564267, ...   1.0 Prediction												

Figura 6.2: Prueba 3

Objeto guardado en la colección "TFGCollectionProcessed" de MongoDB. Como el fallo se ha producido en el proceso del transporte, todos los tanques que fueran en este transporte se han visto afectados.

```
{
    "task": "B",
    "milk tank": "CUIQVGL7N2",
    "minTemperature": -3.595397249360943,
    "maxTemperature": 0.475407672828122,
    "avgTemperature": -1.83137950163396,
    "minHumidity": 60.144005947824986,
    "maxHumidity": 76.657642925539,
    "avgHumidity": 69.13189269448434,
    "prediction": 0,
    "stage": "Prediction"
}
```

---

## **Parte V**

---

# **Conclusiones**

---

## Conclusiones

**E**n este capítulo nombraremos algunos puntos de mejora en el proyecto y la conclusión al finalizar el mismo

### 7.1 Trabajos futuros

Durante el desarrollo de este proyecto ha surgido algunas alternativas o mejoras a la hora de abordar ciertos aspectos del mismo, algunas de estas mejoras son necesarias si deseamos trasladar el proyecto a un entorno de producción y otras que pueden agregar funcionalidad pero que por alcance del trabajo no se han realizado, alguna de estas mejoras son:

- En el desarrollo del dispositivo seria interesante introducir alguna capa de seguridad, encriptando la información enviada por el dispositivo IoT para evitar que un tercero intercepte la comunicación entre estos y pueda obtener cualquier información de la compañía.
- Los prototipos IoT desarrollados tienen funciones muy concretas, por lo que seria conveniente desarrollar estos en sistemas embebidos para mejorar su rendimiento y mantenimiento.
- En el lado del servidor actualmente estamos ejecutando las aplicaciones Spark localmente, esto se podría trasladar y computar en la nube para liberar al servidor de esta carga de trabajo.
- Podríamos añadir vistas extras que requiera el cliente, como la visualización de los últimos procesos pronosticados por Spark
- Seria interesante utilizar herramientas como Docker y Jenkins para automatizar el despliegue del proyecto.

### 7.2 Conclusión

La implementación de todas estas tecnologías y como trabajan conjuntamente no solo ha servido para resolver un problemas concreto como es nuestro caso en la industria láctea, sino que se ha intentado realizar de la forma mas genérica posible para transferir y reutilizar este proyecto en cualquier ámbito de la cadena de suministro. Me ha parecido apasionante como podemos utilizar herramientas como Spark para trabajar con una cantidad ingente de datos y utilizar módulos de aprendizaje automático dentro de la propia herramienta. También personalmente el desarrollo del prototipo Iot ha sido un reto muy interesante por todo lo que este lleva, encontrar los componentes que mas se adapten a mis necesidades e integrar estos en una placa de desarrollo. En conclusión se han alcanzado todos los objetivos propuestos al inicio del proyecto y en especial anticiparnos ante posibles errores de calidad para que este no se propague y se inviertan recursos materiales y económicos innecesariamente.

# Bibliografía

1. Telefónica F. Sociedad Digital en España 2020-2021: El año en que todo cambió [Internet]. Google Books. Fundación Telefónica; 2021 [cited 2022 Nov 26]. Available from: <https://books.google.es/books?hl=es&lr=&id=i34xEAAAQBAJ&oi=fnd&pg=PP24&dq=salesforce+70+gerentes+espa%C3%B1oles&ots=3s5kEbC9Dm&sig=nfN31MgpFPiJMb5JCphNZdu8jVsv=onepage&q=salesforce%2070%20gerentes%20espa%C3%B1oles&f=false>
2. Xu LD, Xu EL, Li L. Industry 4.0: state of the art and future trends. International Journal of Production Research. 2018 Mar 9;56(8):2941–62.
3. Botero C. Mantenimiento preventivo. repositoriosenaeduco [Internet]. 1991; Available from: <https://repositorio.sena.edu.co/handle/11404/1550>
4. LinkedIn. Informe Empleos Emergentes 2020 [Internet]. 21AD. Available from: [https://business.linkedin.com/content/dam/me/business/en-us/talent-solutions/emerging-jobs-report/Emerging\\_Jobs\\_Report\\_112119\\_SP.pdf](https://business.linkedin.com/content/dam/me/business/en-us/talent-solutions/emerging-jobs-report/Emerging_Jobs_Report_112119_SP.pdf)
5. SANZ PINAZO MB. INTOLERANCIA A LA LACTOSA [Internet]. 2019 Jul. Available from: <https://idus.us.es/bitstream/handle/11441/91606/SANZ%20PINAZO%2C%20MAR%C3%8DA%20BEL%C3%89N.pdf?sequence=1&isAllowed=y>
6. Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E. Deep Learning for Computer Vision: A Brief Review [Internet]. Computational Intelligence and Neuroscience. 2018. Available from: <https://www.hindawi.com/journals/cin/2018/7068349/>
7. Kang Y, Cai Z, Tan C-W, Huang Q, Liu H. Natural language processing (NLP) in management research: A literature review. Journal of Management Analytics. 2020 Apr 2;7(2):139–72.
8. BUBER E, DIRI B. Performance Analysis and CPU vs GPU Comparison for Deep Learning [Internet]. IEEE Xplore. 2018. p. 1–6. Available from: [https://ieeexplore.ieee.org/abstract/document/8751930?casa\\_token=p01GMaso\\_00AAAAAA:p-8yrWwAFbC5bov-2I6b9fWS2XWitHnu7u3HEzvlT7s9KEn8DTK36Oi7JzyUzJp0dtK1D6RN](https://ieeexplore.ieee.org/abstract/document/8751930?casa_token=p01GMaso_00AAAAAA:p-8yrWwAFbC5bov-2I6b9fWS2XWitHnu7u3HEzvlT7s9KEn8DTK36Oi7JzyUzJp0dtK1D6RN)
9. PlatformIO. PlatformIO is a professional collaborative platform

- for embedded development [Internet]. PlatformIO. Available from: <https://platformio.org/>
10. Colaboradores de los proyectos Wikimedia. Java: lenguaje de programación [Internet]. Wikipedia.org. Wikimedia Foundation, Inc.; 2003. Available from: [https://es.wikipedia.org/wiki/Java\\_\(lenguaje\\_de\\_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))
  11. Spring. Spring Projects [Internet]. Spring.io. 2019. Available from: <https://spring.io/projects/spring-boot>
  12. Apache Spark. Apache SparkTM - Unified Analytics Engine for Big Data [Internet]. Apache.org. 2019. Available from: <https://spark.apache.org/>
  13. Apache Spark. MLlib | Apache Spark [Internet]. Apache.org. 2019. Available from: <https://spark.apache.org/mllib/>
  14. Apache Spark. Spark SQL & DataFrames | Apache Spark [Internet]. spark.apache.org Available from: <https://spark.apache.org/sql/>
  15. OpenJS Foundation. Node-RED [Internet]. Nodered.org. 2019. Available from: <https://nodered.org/>
  16. MongoDB. ¿Qué es MongoDB? [Internet]. MongoDB. Available from: <https://www.mongodb.com/es/what-is-mongodb>
  17. Banker K, Garrett D, Bakkum P, Verch S. MongoDB in Action: Covers MongoDB version 3.0 [Internet]. Google Books. Simon and Schuster; 2016 [cited 2022 Nov 26]. Available from: <https://books.google.es/books?hl=es&lr=&id=kzkzEAAAQBAJ&oi=fnd&pg=PT21&dq=mongodb+examples&ots=8U60qX6199&sig=GPdJwKEMcwO4IdKnUZ0SZXYP0Rsv=onepage&q=mongodb%20examples&f=false>
  18. Ionic. Ionic - Cross-Platform Mobile App Development [Internet]. Ionic Framework. 2019. Available from: <https://ionicframework.com/>
  19. Angulo R. Aplicaciones móviles híbridas: lo mejor de dos mundos TECNOLOGÍA MÓVIL [Internet]. Available from: <https://cmappspublic2.ihmc.us/rid=1NTQ9NMKD-R1SKBP-24M4/Aplicaciones%20moviles%20hibridas-%20lo%20mejor%20de%20dos%20mundos.pdf>
  20. Fritzing. Fritzing [Internet]. fritzing.org. Available from: <https://fritzing.org/>
  21. Arduino. Arduino - SoftwareSerial [Internet]. Arduino.cc. 2019. Available from: <https://www.arduino.cc/en/Reference/softwareSerial>
  22. Arduino. Adding More Serial Ports to your board. | Arduino Documentation [Internet]. docs.arduino.cc. Available from: <https://docs.arduino.cc/tutorials/communication/SoftwareSerialExample>

23. Hart M. mikalhart/TinyGPS [Internet]. GitHub. 2022 [cited 2022 Nov 26]. Available from: <https://github.com/mikalhart/TinyGPS>
24. TinyGPS. TinyGPS | Arduiniana [Internet]. Available from: <http://ardiniana.org/libraries/tinygps/>
25. "Extract GPS location in Arduino with NEO-6m module", GPS Neo 6m, May 2019, [online] Available: <https://www.youtube.com/watch?v=rCXYP6H3gSE>.
26. Google. Google Maps Platform | Google Developers [Internet]. Google Developers. 2019. Available from: <https://developers.google.com/maps/documentation/>
27. SIM Com. SIM800L, SIM800L Datasheet, SIM800L PDF - SIM Com [Internet]. datasheet-pdf.com. [cited 2022 Nov 26]. Available from: <https://datasheet-pdf.com/PDF/SIM800L-Datasheet-SIMCom-989664>
28. "What is GSM Module?", GSM, May 2019, [online] Available: <https://www.engineersgarage.com/what-is-gsm-gprs-module>.
29. Babiuch M, Foltynek P, Smutny P. Using the ESP32 Microcontroller for Data Processing. 2019 20th International Carpathian Control Conference (ICCC). 2019 May;
30. Maier A, Sharp A, Vagapov Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things [Internet]. IEEE Xplore. 2017. p. 143–8. Available from: <https://ieeexplore.ieee.org/abstract/document/8101926/>
31. Hao Yu Electronics. Temperature and humidity module AM2301 Product Manual [Internet]. Available from: <http://www.haoyuelectronics.com/Attachment/AM2301/AM2301.pdf>