Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

# Aplicación web para la gestión de almacén y partes de trabajo de una empresa de tecnología

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: Sergi Vendrell García

**Tutor**: Andrés Boza García

2019-2020

# Resumen

El objetivo de este trabajo es la optimización de los procesos de almacenaje de productos y creación de partes de trabajo mediante la creación de una herramienta informática profesional y completa. Para ello se han estudiado los procesos relativos a estos dos sistemas para encontrar los problemas que tenía la metodología antigua y finalmente ofrecer una más eficiente. Partiendo de esta optimización en los procesos se ha realizado una aplicación web que permite informatizar y automatizar la realización de estos procesos.

**Palabras clave:** optimización, almacenaje, procesos, aplicación, automatizar.

# Abstract

The aim of this work is to optimise product storage processes and the creation of working parts by creating a professional and complete computer tool. For this purpose, the processes related to these two systems have been studied in order to find the problems that the old methodology had and finally to offer a more efficient one. Based on this optimisation of the processes, a web application has been created to computerise and automate these processes.

**Keywords :** optimise, storage, processes, application, automate.

# Tabla de contenidos

# Tabla de figuras

# 1. Introducción

Debido a diversos factores como el aumento del flujo de comercio a nivel internacional, el auge imparable de las transacciones online y el crecimiento de las grandes empresas, se han presentado problemáticas respecto al almacenaje, gestión y logística de los productos o materiales con los que trabaja una organización.

Para solventar los problemas de logística derivados de los factores anteriores, surgen los sistemas de gestión de almacenes o SGA. Podemos definir un SGA como una herramienta informática que permite a las empresas optimizar todos los procesos relacionados con su almacén y, con ello convertirlos en más eficientes (OBS Business School, 2020). Fundamentalmente, podemos dividir los procesos que gestiona un SGA en cuatro bloques:

- Control de entradas y salidas
- Control de stock
- Control de ubicaciones y disposición
- Funcionalidades personalizadas a cada almacén

## 1.1. Motivación

A pesar de que este tipo de aplicaciones llevan siendo utilizadas desde hace muchos años por las grandes empresas, siempre admiten margen de mejora ya que la tendencia por almacenes automatizados sigue creciendo y sigue siendo común encontrarse con la reticencia de las pequeñas y medianas empresas a instalar sistemas de este tipo en sus instalaciones.

Se pretende poner en duda la creencia de que el coste de desarrollo de estas soluciones no llega a compensar el aporte que generan a la empresa. Algunas empresas rechazan directamente estas soluciones sin tener en cuenta que, en muchas ocasiones la cantidad de recursos que se pueden aprovechar puede mejorar su capacidad de producción y la utilización de recursos.

La motivación detrás de este trabajo de fin de grado reside en ofrecer soluciones escalables de calidad para gestionar y automatizar tareas que pueden resultar complejas para las empresas, con la finalidad de ser una herramienta más en su día a día para una gestión óptima.

Por otra parte, con este proyecto, el autor busca consolidar los conocimientos adquiridos durante el grado de una manera profesional y disfrutar del complicado y emocionante proceso de aprendizaje que supone trabajar con elementos desconocidos.

## 1.2. Objetivos

- Simplificar los procesos realizados por los empleados para la gestión de productos.
- Reducir los tiempos utilizados en las operaciones que se realizan dentro de las instalaciones.
- Realizar una aplicación web lo más profesional posible y teniendo siempre en mente la escalabilidad de esta.
- Documentar la aplicación desde el inicio, durante y posteriormente al desarrollo de esta.
- Informatizar los partes de trabajo que realizan los empleados.
- Aumentar la eficiencia del espacio utilizado para el almacenamiento de material.
- Estudiar el impacto que puede llegar a tener el desarrollo de una aplicación de este tipo para una empresa.
- Simplificar el proceso de realización de los partes, agilizando su creación.
- Ofrecer en cualquier instante la situación actual del almacén, sus productos y ubicaciones.

## 1.3. Estructura

La estructura de esta memoria está formulada para que se pueda apreciar el proceso de su creación. Este trabajo se divide en tres grandes bloques, el primer bloque comprende los capítulos 1 y 2. En este bloque se va a realizar una contextualización de las herramientas que se utilizan actualmente en el sector profesional para producir aplicaciones similares a la propuesta en el proyecto. Este capítulo recibirá el nombre de estado del arte y contendrá toda la información actual necesaria para elaborar dicha aplicación, así como también algunos debates que se han generado sobre la seguridad de estas según las necesidades de cada compañía.

El segundo bloque comprende los capítulos 3, 4 y 5. En este bloque se realizará un análisis del problema, estudiando todas sus las variables con la única finalidad de adecuar, en la medida de lo posible, nuestra solución a las necesidades de la empresa. Tras determinar la magnitud del problema, se realizará un capítulo que recopila la propuesta de solución a la empresa teniendo en cuenta el estado del arte y el análisis del problema explicados en los capítulos que lo preceden.

Por último, el último bloque comprende los capítulos 6, 7 y 8. Este bloque consistirá en documentar el desarrollo de la aplicación, partiendo de la capa más profunda de la aplicación o lado del servidor, para poco a poco ir ascendiendo a la más cercana al usuario, en este caso la interfaz. Por otro lado, también se examinará la aplicación para comprobar que se adecua a los objetivos y exigencias de la empresa. Por último y para concluir el trabajo, se redactará una conclusión mostrando todos los resultados obtenidos durante y posteriormente al desarrollo de la aplicación y haciendo balance de la utilidad que este tipo de aplicaciones supone para las empresas.

## 1.4. Colaboraciones

Es conveniente mencionar que este trabajo de fin de grado será realizado bajo la supervisión, opinión y colaboración de algunos empleados de DISOLTEC, una empresa con mucha experiencia en el sector de la automoción que actualmente trabaja codo con codo con empresas como Ford o Renault entre otras, desarrollando aplicaciones para gestionar sus almacenes y servicios.

A pesar de que, inicialmente, es el alumno quien hospedará el proyecto en su casa, se prevé que, una vez finalizado y cuando la situación lo permita, se trasladará a la oficina de DISOLTEC. Así pues, el sistema de gestión de almacén caótico desarrollado en este trabajo de fin de grado será, al fin y al cabo, una herramienta profesional de uso en el día a día de la gestión de la empresa.

Cabe destacar que, DISOLTEC prestará instalaciones tangibles y formación que permitirán el desarrollo y la implementación del proyecto. El principal motivo de su colaboración es aplicar su amplia experiencia en el campo de desarrollo de aplicaciones para gestión de almacenes y desarrollar una solución lo más personalizada y profesional posible.

# 2. Estado del arte

## 2.1. Aplicaciones de gestión de almacenes

Para empezar con este trabajo de fin de grado, lo primero que se va a realizar es una investigación sobre aplicaciones similares que se encuentren en el mercado y sus distribuidores. Pueden ser de interés factores como quién desarrolla este tipo de soluciones, cómo las desarrollan, qué tipo de aplicación utilizan, entre otras. De esta forma, se logrará focalizar mejor los esfuerzos de esta investigación hacia la dirección correcta.

En la actualidad, existe un gran número de empresas de tecnología dedicadas a desarrollar y personalizar aplicaciones de gestión de almacenes a todo tipo de empresas, adaptándose tanto a la mercancía que tienen como a cualquier otra necesidad que presenten. Es por eso que, en este apartado se ha decidido analizar tres herramientas diferentes con una personalización adaptable a cualquier tipo de empresa y desarrolladas por referentes en el sector.

**Mecalux – Easy MWS**

Mecalux es una de las compañías punteras en el mercado de sistemas de almacenaje. Su actividad consiste en el diseño, fabricación, comercialización y prestación de servicios relacionados con las estanterías metálicas, almacenes automáticos y otras soluciones de almacenamiento. Compañía líder en España, se sitúa en el tercer puesto mundial en el ranking de su sector, con ventas en más de 70 países. (Mecalux, 2020)

Mecalux proporciona a las empresas con todos los elementos necesarios para la correcta gestión de un almacén, desde estanterías metálicas adaptadas a los productos, pasando por PDAs lectoras de etiquetas y también un software de gestión del propio almacén que permite su completa gestión y automatización mediante algoritmos.

Centrándonos en el software que ofertan en su propia página web, se extrae que comercializan una aplicación basada en dos modalidades, la modalidad SaaS (Software as a Service) centrada en la distribución en línea de una aplicación en la nube o bien la modalidad on-premise basada en que la aplicación y hardware requeridos para hacer funcionar el sistema se encuentran alojadas en la instalación del cliente. Mecalux trabaja para clientes como Porcelanosa, DHL, Danone, Michelin, Leroy Merlin y Nestlé entre otras.

**Datadec – expert SGA**

Datadec lleva más de 30 años ofreciendo aplicaciones y servicios empresariales a la mediana y grande empresa. Son los responsables de diseñar y aplicar soluciones que marquen la diferencia operativa de los clientes. Su misión es ofrecer un servicio post-venta excelente que les permita garantizar la continuidad del negocio y que junto con las nuevas tecnologías disruptivas que aplican, sus clientes puedan monetizar las inversiones realizadas. (Datadec, 2020)

Datatec comercializa una aplicación de gestión de almacenes llamada expert SGA totalmente personalizable y configurable para cada empresa, esta aplicación es desplegada y utilizada por el cliente mediante un servicio web que ofrece una alta escalabilidad, disponibilidad e integración con terceros.

Según ellos, esta aplicación cuenta con funcionalidades como definir todo tipo de tareas, definir zonas de almacén, asignación de recursos y creación de una estructura de operarios en el almacén. Dicen contar con más de 50 algoritmos de entradas, aunque no se especifica bien cuales y alrededor de 70 algoritmos de salidas los cuales tampoco vienen detallados.

### SCM Logistica

Se definen como un equipo de profesionales experimentados en diferentes ámbitos, haciendo uso de conocimientos especializados en función de las necesidades del cliente. Cuentan con más de 20 años de experiencia en el campo de almacenaje y la distribución. Su experiencia abarca desde pequeñas y medianas empresas hasta grandes multinacionales dedicadas a distribución, alimentación, químico, farmacéutico, producción, retail. Se identifican con la calidad de sus soluciones y su diferencial visión de equipo respecto a sus competidores. (SCM Logística, 2020)

En su página web no especifican muy bien cómo esta implementado su software, no obstante, por los vídeos que tienen publicados en su canal de Youtube parece que ofrecen una aplicación de escritorio con una cantidad de funcionalidades predefinidas de manera genérica para todos sus clientes.

Algunas de las características que destacan son el control de los movimientos que se producen en el almacén en tiempo real, preparación y recepción de pedidos utilizando WiFi o radiofrecuencia, explotación y consultas de datos mediante distintos informes navegables y generables, entre otras. Cuentan con múltiples estrategias de picking como FIFO, LIFO o FEFO.

## Conclusiones

Teniendo en cuenta las soluciones que ofrecen estas empresas, se extrae que decantarse por una aplicación web es, en términos generales, una decisión más útil e interesante. Las aplicaciones web, a diferencia de las aplicaciones de escritorio, proporcionan mucha más escalabilidad, personalización y disponibilidad ya que son aplicaciones que no se encuentran alojadas en las instalaciones del cliente.

Por otra parte, si tenemos en cuenta que estas aplicaciones van a ser utilizadas en un entorno de producción, el cliente nos va a exigir una gran disponibilidad; utilizando aplicaciones web tenemos todos los recursos disponibles en línea, esto significa que la aplicación puede ser accedida desde cualquier dispositivo que cuente con un navegador y conexión a internet o intranet sin necesidad de malgastar tiempo instalando los programas necesarios para ejecutar una aplicación de escritorio.

Por estos motivos y porque sería muy extenso abarcar el contexto de aplicaciones web y aplicaciones de escritorio de manera detallada, se decide a partir de este punto enfocar los esfuerzos de la contextualización hacía las aplicaciones y servicios web en la actualidad.

## 2.2. Aplicaciones web

Antes de meternos de lleno en el proyecto, vamos a realizar una breve introducción a la situación actual en el desarrollo de aplicaciones web con características similares a la que ocupa este trabajo de fin de grado.

Debido a que existen una inmensa cantidad de sinergias con herramientas para la creación de una aplicación web, sólo vamos a centrarnos en los aspectos que consideramos más importantes para que resulte lo más completa y compacta posible.

Siguiendo con lo expuesto en los párrafos anteriores, la contextualización de las aplicaciones web irá desde la parte más cercana al usuario como es la interfaz, pasando por la parte intermediaria entre interfaz y sistema y, por último, se hablará de la parte del servidor, que recibe peticiones de la aplicación intermediaria de las aplicaciones web.

### 2.2.1. Contexto de la metodología frontend

Podemos definir frontend como la parte de nuestra aplicación con la que el usuario tiene contacto, interactúa y visualiza. Mediante el frontend, el usuario debe ser capaz de entender el funcionamiento de nuestra aplicación y ser capaz de llevar a cabo las actividades deseadas.

Hay muchas maneras de dividir el contenido que engloba el frontend, pero en este caso lo vamos a dividir en tres secciones. Por una parte, tenemos la estructura, en la que se trabaja el dónde y cómo se organizan los elementos con los que va a interactuar el usuario.

Por otra parte, necesitamos dar formato y aspecto a la estructura que ofrecemos al cliente. Para cumplir con este propósito existe la presentación, que nos será de gran utilidad para ofrecer un aspecto atractivo y útil a nuestro usuario. Por último, hablaremos de la interactividad, que es la encargada de la parte lógica del frontend o el lado del cliente. En ella se pueden realizar ciertos cálculos y funciones para mostrar al usuario.

Este tipo de estructura posee muchas ventajas, algunas de ellas son que nos permite realizar cambios prácticamente instantáneos en la parte externa, reutilizar código de la parte intermedia y efectuar la depuración o modificación de alguna capa sin afectar a las otras.

Habiendo completado esta primera toma de contacto, vamos a profundizar en las herramientas más utilizadas y sus características en el contexto actual durante la realización de este trabajo de fin de grado.

#### *2.2.1.1. Estructura*

En lo que concierne a la estructura en la creación de una página web, la voz de los desarrolladores es casi unánime. La herramienta más utilizada y globalizada es HyperText Markup Language, más conocida como HTML en sus diferentes variantes y versiones. HTML es una herramienta sencilla, intuitiva y potente, por lo que es la preferida entre los desarrolladores frontend para trabajar con la estructura de las aplicaciones web.

Parafraseando a (Brooks, 2007), HTML es un lenguaje que consiste esencialmente en dos partes: el contenido o información y unas instrucciones llamadas etiquetas, que le dicen al navegador cómo mostrar el contenido en forma de página web. De este modo, más que un lenguaje podríamos considerar a HTML como un conjunto de instrucciones que interpreta nuestro navegador.

Estas etiquetas son tan potentes que podemos crear todo tipo de estructuras imaginables en una página web con ellas, desde un simple título hasta tablas gigantes que albergan miles y miles de celdas de información extraídas de la base de datos situada en el backend o lado del servidor. Esto convierte a HTML en una herramienta muy potente que nos permite, desde realizar estructuras muy simples orientadas a aplicaciones livianas, a trabajar con otras realmente complejas y más pesadas.

Una vez hecha esta introducción a HTML, vamos a entender un poco mejor por qué este lenguaje es tan ampliamente utilizado en el desarrollo de la estructura de una aplicación web. Para ello vamos a ver algunas ventajas e inconvenientes de HTML basándonos en los conceptos expuestos en (EDUCBA, no date). Empezando por las ventajas.

- Es un lenguaje que podemos utilizar sin necesidad de ningún software de terceros y sin ningún tipo de coste.
- Nos permite una edición muy rápida y eficaz de la página web. Basta con cambiar un par de etiquetas y recargar la página para ver los cambios que acabamos de realizar.
- Una de las razones del éxito de HTML reside en la facilidad que presenta para ser integrado en otros lenguajes como puede ser Javascript, PHP, node.js y CSS entre muchos otros. Esto nos abre un abanico de posibilidades muy amplio para trabajar con HTML.
- Otra de las ventajas la hemos mencionado antes en los párrafos anteriores: HTML es un lenguaje mundialmente utilizado, lo cual hace que exista una cantidad ingente de documentación y usuarios dispuestos a ayudarnos en cualquier duda que nos surja y que podemos usar cómo guía en nuestro desarrollo.
- La mayoría de los navegadores lo soportan. Esto le otorga mucha versatilidad ya que nos permite desarrollar aplicaciones multiplataforma sin tener que escribir código extra.

Pero no todo son ventajas. Si vamos a elegir HTML como nuestro editor para estructuras, debemos tener en cuenta que tendremos que lidiar con algunas limitaciones. Estas son las más importantes:

- HTML requiere de mucho código para crear una simple página web y, aunque no es nada complejo, a veces puede resultar tedioso. Sobre todo cuando ya contamos con una estructura diseñada y tenemos que depurarla.
- No está centralizado, es decir, si tuviéramos que trabajar con varias pestañas en una página web, tendríamos que utilizar distintos ficheros para modificarlos y no podríamos realizar cambios en otras ventanas desde un fichero.
- Citando a (Brooks, 2007), idealmente, HTML debería funcionar de la misma manera en todos los navegadores, pero, no obstante, sólo se consigue una aproximación a este concepto en la práctica.

#### *2.2.1.2. Presentación*

Después de estudiar cómo desarrollar la presentación de una aplicación web, llegamos a la conclusión de que, a pesar de que existen distintas herramientas para desarrollar la presentación de una aplicación web, la mayoría de ellas están basadas en Cascade Style Sheet o su abreviatura CSS. Por ello mismo, sólo se va a profundizar en CSS, ya que es una herramienta completa y útil para este propósito.

Cascade style sheet es un lenguaje ampliamente utilizado para modificar la presentación del contenido de documentos por ejemplo HTML. De esta manera CSS se utiliza normalmente como una herramienta complementaria a HTML para dar aspecto o presentación a la estructura de una aplicación.

Parafraseando el libro (Duckett, 2011), nos permite crear reglas que especifican cómo debe aparecer el contenido de nuestros elementos. Esto nos va a permitir desarrollar páginas web más atractivas y controlar su diseño. Su funcionamiento es simple; cuando utilizamos un navegador, este lee el documento HTML para construir la página web y la información de la hoja de estilo para dar el aspecto deseado a la página.

Una vez completada esta breve introducción a esta herramienta, vamos a ver directamente sus ventajas y desventajas parafraseando el artículo de (Sahu, no date). Vamos a empezar por las ventajas:

- Permite crear estilos predefinidos que pueden ser aplicados en cualquier parte del documento HTML, esto nos permite ahorrar mucho tiempo en dar aspecto a nuestra aplicación web, ya que un mismo estilo puede ser reutilizado en diferentes elementos.
- Requiere muy poco código para hacer grandes cambios visuales. Esto impacta directamente al rendimiento y la eficiencia de nuestra aplicación, cuanto más ligera más rápido cargará y mejor experiencia tendrá el usuario en nuestra aplicación web.
- Permite reposicionar elementos de la capa de estructura de manera sencilla.

Ya mencionados los aspectos negativos del uso de CSS en nuestra aplicación web, ahora vamos a echar un vistazo a las limitaciones que posee:

- No todos los navegadores leen de la misma manera CSS, lo que puede provocar que nuestro código sea programado y funcione correctamente en un navegador especifico pero que no se visualice bien en otros.
- A pesar de que CSS está en actualizaciones constantes, sigue habiendo ciertos elementos que no interaccionan como deberían y nos pueden traer algún que otro dolor de cabeza en situaciones específicas.

#### *2.2.1.3. Interactividad*

Si estudiamos la interactividad a lo largo de los años, no tardaremos en darnos cuenta de que Javascript es el lenguaje a tener en cuenta para el desempeño de esta capa, no obstante, podemos ver que existe un ecosistema muy grande alrededor de Javascript y que es más importante la elección de un framework adecuado para nuestra aplicación web.

Según (McFarland, 2008), hace no mucho tiempo, internet era un lugar plagado de páginas web con diseños planos HTML, las páginas web sólo mostraban información estática. Javascript es un lenguaje de programación que nos permite cargar a nuestros documentos HTML de animación, interactividad y diferentes efectos visuales dinámicos.

Actualmente, la mayoría de las webs que conocemos responden al usuario como si fueran aplicaciones normales de escritorio, reaccionando de manera inmediata a cada click que se realiza. Todo esto es gracias la aparición de Javascript como lenguaje para manejar la lógica de las aplicaciones webs.

Si nos remontamos a los inicios del desarrollo de aplicaciones web, se puede ver que en las primeras aplicaciones se utilizaba sólo Javascript. Con el paso de los años han ido apareciendo distintos frameworks cuyo único propósito es ofrecer funcionalidades y facilitar la programación en este lenguaje.

Hoy en día existen muchas opciones entre las que elegir, así que en este contexto vamos a hacer hincapié en las que consideramos más relevantes. Para este ejercicio de contexto hemos elegido distintos frameworks o herramientas atendiendo a los más utilizados e innovadores del momento.

**Angular JS**

Basándonos en la propia documentación de (AngularJS, no date), Angular es un framework de Javascript de código abierto creado y mantenido por Google. Actualmente se utiliza para la lógica de la interacción del usuario con el frontend o interfaz. Está basado en la estructura modelo, vista, controlador o MVC.

Es importante recalcar que Angular es un framework cuyo único propósito es el desarrollo de la lógica y funcionalidad de aplicaciones web. No fue concebido para desarrollar páginas más estéticas ni aporta nada en ese contexto.

Seguidamente, vamos a ver algunas ventajas que podemos obtener si hacemos de Angular nuestro principal framework para trabajar con Javascript:

- Una de las principales ventajas de Angular es que está desarrollado en TypeScript, lo cual aporta mucha consistencia y legibilidad a las aplicaciones web. Esto puede ser una ventaja o desventaja dependiendo del desarrollador. No obstante, Angular no nos fuerza a programar en Typescript, ya que existen alternativas.
- Facilidad para el mantenimiento de aplicaciones web ya que aporta funcionalidades a Javascript, lenguaje que ya de por sí no es complejo de depurar y en ocasiones nos agiliza el trabajo.
- Angular cuenta con un sistema de componentes web. Esto nos permite reutilizar esos componentes en otro tipo de aplicaciones si fuera necesario. De esta manera, nos podemos ahorrar líneas de código, haciendo que nuestra aplicación sea más legible y eficiente.
- Angular pretende ser una apuesta de futuro que sea utilizable a largo plazo, ya que Javascript sufre muchos cambios y novedades en cortos periodos de tiempo.

A continuación, enumeraremos algunas de las desventajas más relevantes que presenta Angular y que sin duda debemos tener en cuenta antes de seleccionarlo como nuestro principal framework de trabajo.

- Si se quisiera migrar a las últimas versiones de Angular, es importante tener en cuenta que podrían surgir muchos problemas debido a la complejidad que supone migrar código a distintas versiones de Angular.
- A pesar de que Angular fue diseñado como una propuesta a largo plazo, han sacado distintas versiones en las que se han realizado grandes cambios que en muchas ocasiones no han contentado a los desarrolladores.

**ReactJS**

Siguiendo con las herramientas para gestionar la lógica de nuestra aplicación, es el turno de ReactJS, una biblioteca Javascript cuyo objetivo es facilitar la creación de aplicaciones web de una sola página.

React es una librería de Javascript creada y gestionada por Facebook para crear interfaces interactivas orientadas al usuario. React se utiliza principalmente para crear y reutilizar potentes componentes. Probablemente la característica más valorada de React es que puede ser utilizado en el cliente, servidor, aplicaciones móviles e incluso aplicaciones de realidad virtual (Santana Roldan, 2018).

Veamos qué beneficios diferencian a ReactJS de las bibliotecas o frameworks descritas anteriormente utilizando conceptos de (Schwarzmüller, 2019).

- En ReactJS podemos utilizar JSX, una herramienta que no tiene definición semántica y que es parecida a XML. Esto convierte a React en una librería muy útil, pues nos permite utilizar etiquetas HTML dentro de Javascript. Para ilustrar esto mejor, se podría decir que JSX es una extensión de la sintaxis de Javascript.
- A diferencia de otras bibliotecas o frameworks vistas con anterioridad, ReactJS tiene un flujo de datos unidireccional. Los componentes superiores propagan datos a los componentes inferiores. Si cambia el estado, se vuelven a actualizar los componentes superiores en cascada.
- Una de las principales características de React es el Virtual DOM. Antes de entender esta ventaja necesitamos saber que se le llama DOM a la estructura de objetos que genera un navegador cuando carga un documento. Esto es muy conveniente porque cuando actualizamos una vista, React actualiza nuestro Virtual DOM en lugar de actualizar el DOM del navegador. De esta manera cuando React compara el Virtual DOM y el DOM del navegador sabe si ha cambiado algo, lo que nos proporciona rapidez y eficiencia.

### Vue.js

Una vez vistos AngularJS y ReactJS vamos a pasar a analizar qué nos ofrece Vue.js en el contexto de creación de aplicaciones web. Citando a (Schwarzmüller, 2019), Vue.js es un framework Javascript de código abierto creado por Evan You en 2014, el objetivo con el que empezó fue de ser un framework con las mejores funcionalidades de Angular, pero más ligero y eficiente.

Vamos a ver por qué Vue.js se ha ganado un espacio entre los mejores frameworks del momento y qué nos ofrece actualmente. Parafraseando el artículo escrito en Genbeta (Dongil Sánchez, 2018).

- Vue.js está modularizado. Esto simplifica el framework y lo hace más sencillo de aprender. Conforme vayamos haciéndonos con Vue podremos ir añadiendo más módulos y funcionalidades para hacer nuestra aplicación web más completa.
- Una de las cosas que los programadores alaban de Vue es el ecosistema que lo rodea. Estas herramientas permiten al programador entender en todo momento qué está haciendo y cómo se está llevando a cabo.
- Cabe destacar también que el equipo de desarrollo de Vue cuenta con una extensión de Chrome que nos permite visualizar cómo se está renderizando nuestro árbol de componentes. Esto es muy útil ya que nos permite ver cómo se están lanzando y registrando los eventos.
- Como otros de los frameworks o bibliotecas que hemos visto con anterioridad, Vue también nos permite tener todos los componentes en un mismo fichero, lo cual nos da mucha flexibilidad y comodidad a la hora de realizar cambios en nuestra aplicación web.

### 2.2.2. Contexto de la metodología backend

Si bien anteriormente hemos descrito el frontend como todo aquello con lo que se relaciona o interactúa el usuario, el backend podría ser definido como todo lo contrario. Actualmente entendemos por backend de una aplicación a todo aquello a lo que el usuario no puede acceder o relacionarse de manera directa.

Podríamos decir que el backend comprende toda aquella lógica de la aplicación que el usuario no ve, pero es imprescindible para que pueda interactuar correctamente con el frontend o lado del usuario.

A diferencia del frontend, el backend tiene una gran cantidad de lenguajes y herramientas con las que trabajar. En este trabajo agruparemos el backend en lenguajes, bases de datos y servidores web teniendo en cuenta el actual contexto profesional de las aplicaciones y servicios web.

#### *2.2.2.1. Contexto de las APIs*

Para entender el contexto actual de las APIs en el sector profesional, es conveniente que primero entendamos qué es una API y el papel que desempeña dentro de una aplicación o en este caso, dentro de una aplicación web.

La palabra API significa interfaz de programación de aplicaciones. Básicamente una API permite que los productos y servicios se comuniquen con otros sin necesidad de saber cómo están implementadas las otras partes. Esta imagen muestra un esquema visual en el que podemos ver el papel de la API en una aplicación web.

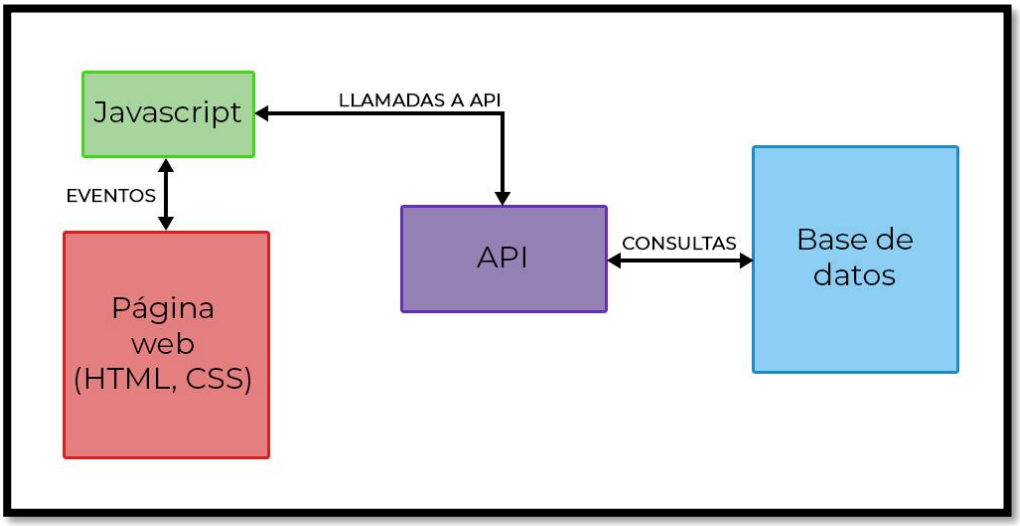



*Figura 1. Estructura de una aplicación web*

Cada API está definida en un lenguaje de programación concreto y con especificaciones y estructuras distintas en cada caso. Es por eso por lo que actualmente existen muchos tipos de APIs. No obstante, ya que este trabajo de fin de grado decide profundizar sobre las aplicaciones web, sólo desarrollaremos las APIs que pueden ser utilizadas con el fin de ofrecer servicios web.

En el contexto de APIs para servicios web, todo el intercambio de información se realiza normalmente mediante intercambios de peticiones HTTP o HTTPS dependiendo del tipo de página web. Para ello se utilizan POST y GET, dos métodos muy extendidos en la comunicación entre las diferentes capas de una aplicación web.

Una vez aclarado esto vamos a plantear dos tipos de APIs de servicio web a modo de ejemplo de entre todas las que existen, en este caso, vamos a hablar de las Soap API y las RestAPI y veremos las ventajas de Soap sobre Rest y viceversa.

Antes de hacer la comparación, es importante tener en cuenta que estas pueden ser ventajas o desventajas según esté orientado nuestro proyecto y, por tanto, algo considerado una desventaja puede resultar una ventaja en algún caso concreto. Dicho esto, vamos a analizarlas un poco más en profundidad.

Utilizando los conceptos de (SoapUI, no date) vamos a recopilar y hablar sobre algunas de las principales diferencias entre APIs de tipo Rest y Soap.

Estas son algunas de las ventajas que nos ofrece Soap respecto a Rest:

- Mayor independencia de lenguaje, plataforma y transporte debido a que Rest requiere que utilicemos el protocolo HTTP en todas las comunicaciones, mientras que Soap es más flexible.
- Funciona bien en entornos distribuidos.
- Es un tipo de interfaz de programación de aplicaciones que está muy estandarizada en el sector.
- Nos puede proporcionar automatización si la utilizamos con un lenguaje que lo permita.

Por otra parte, aquí presentamos algunas ventajas de Rest respecto a Soap:

- Utiliza estándares muy fáciles de entender como Open API Specification.
- Más sencillo de aprender y dominar que Soap.
- Más eficientes que Soap en lo que a comunicación respecta. Mientras Soap emplea XML para todos sus mensajes, Rest emplea normalmente JSON.
- Las aplicaciones Rest permiten una comunicación más ágil y por tanto permiten acelerar las respuestas al cliente.

### *2.2.2.2. Lenguajes y frameworks*

En la actualidad existen muchos lenguajes, frameworks y librerías con los que trabajar en el backend. En este apartado vamos a ver las herramientas más utilizadas en la programación en el lado del servidor o backend. Veremos qué nos aportan, sus ventajas, inconvenientes y recalcaremos, si procede, algún detalle interesante que pudiera ser útil en el desarrollo backend de una aplicación o servicio web.

Hemos seleccionado diferentes lenguajes y frameworks teniendo en cuenta la convención de los programadores en el actual sector profesional y sin dejar de lado la configuración y las características que estos nos pueden aportar. Estas son las herramientas seleccionadas que vamos a analizar con más profundidad:

#### ***Ruby on Rails***

Para entender un poco qué es Ruby on rails, de dónde surge este lenguaje y su complementario framework, acudimos a un libro muy recomendable que explica Ruby on rails desde cero. (Hartl, 2012) Ruby on rails (o sólo "Rails" para abreviar) es un framework de desarrollo web escrito en el lenguaje de programación Ruby. Desde su lanzamiento en 2004, Ruby on Rails se ha convertido rápidamente en una de las herramientas más potentes y populares para construir aplicaciones web dinámicas.

Algunas de las compañías más famosas que utilizan Ruby on Rails son Airbnb, Basecamp, Disney, Github, Shopify y Twitter entre muchas otras.

Ruby on Rails se ha vuelto tan popular por muchas razones. Seguidamente vamos a realizar un análisis para explicar por qué se ha convertido en un framework ampliamente utilizado en la actualidad:

- Ruby on rails es un software de código abierto y por ello mismo su descarga es totalmente gratuita. Debe mucho éxito a su aspecto y diseño compacto gracias a Ruby, mientras Rails es una herramienta muy útil para desarrollar aplicaciones web.
- Rails fue uno de los primero frameworks en implementar la arquitectura Rest para estructurar aplicaciones web.
- Una ventaja importante de Ruby on Rails es la comunidad que tiene. Debido a que es un software de código abierto, es muy sencillo encontrar ejemplos de código en foros y usuarios dispuestos a ayudar en caso de dudas.

#### ***Django con Python***

(Dauzon, Bendoraitis and Ravindran, 2016) Django nace en 2003 en Lawrence, Kansas. Es un framework para el desarrollo web que utiliza Python como lenguaje para la elaboración de páginas web. Su objetivo es escribir webs dinámicas y rápidas. En 2005, la agencia creadora decidió publicar el código de Django bajo una licencia BSD. En 2008 se fundó Django Software Foundation para dar soporte a Django.

Actualmente, existen muchas empresas que trabajan con Django y Python. Algunas de ellas son Instagram, Pinterest, Udemy y Trivago. Con el tiempo se ha vuelto un framework muy popular en entornos de desarrollo web. Estas son algunas de las principales características que nos ofrece la sinergia de Django con Python:

- Como hemos mencionado antes, Django está publicado bajo una licencia BSD lo que permite que sea usado de manera gratuita por todo el mundo, lo cual también es algo a tener en cuenta al elegir una herramienta backend.
- Django posee un sistema de módulos que permite al desarrollador customizarlo al máximo. En muchos casos esto será muy útil para el desarrollador debido a la gran calidad que tienen los módulos.
- Mezclar Django con Python nos permite utilizar todos los beneficios mencionados anteriormente y todas las librerías y sencillez de Python. Esto los convierte en un tándem muy interesante para el desarrollo de aplicaciones web.
- Nos aporta un código simple y legible. La filosofía de Django es intentar alcanzar la perfección en el código. Es por esto por lo que se esfuerzan mucho en ofrecer una sintaxis simple y sencilla de entender que puede ahorrar al programador muchos dolores de cabeza depurando código.

### *ASP.NET Core*

(Ragupathi, De Sanctis and Singleton, 2017) ASP.NET Core es la última versión de ASP.NET MVC publicada por Microsoft. Se trata de un framework de desarrollo web en la parte del servidor que nos ayuda a construir aplicaciones web de manera efectiva. Funciona con ASP.NET 5, que nos permite ejecutar nuestra aplicación en una gran variedad de plataformas incluyendo Linux y MacOS.

Hoy en día, ASP.NET Core es un framework que está muy extendido en empresas grandes, medianas y pequeñas. Estas son algunas de las particularidades que han hecho que tanto ASP.NET como su versión Core estén tan extendidas actualmente en el sector profesional del desarrollo de servicios web:

- ASP.NET Core ofrece un entorno en el que se puede crear una API web. No obstante, también permite crear interfaces de usuario web.
- Capacidad de desarrollo multiplataforma. ASP.NET Core puede ser desarrollado y ejecutado en Windows, MacOS y Linux.
- Es una herramienta de código abierto, gratuita y centrada en su extensa comunidad de desarrolladores.
- ASP.NET Core está preparada para trabajar con servicios de datos en la nube.
- Permite hospedar páginas web en Kestrel, IIS, HTTP.sys, Nginx, Apache y Docker y cuenta con herramientas que simplifican el desarrollo web moderno.

### *Hypertext Preprocessor PHP*

(Stobart, 2004) PHP (un acrónimo de 'PHP Hypertext Preprocessor') es un lenguaje de código abierto diseñado para crear webs dinámicas para aplicaciones web. Estas páginas web no permanecen inmóviles, sino que interactúan con el usuario, consiguiendo una experiencia más rica e interesante con el servicio web.

A pesar de que PHP es un lenguaje que se publicó en 1995, sigue siendo en la actualidad uno de los lenguajes preferidos por los programadores para desarrollar servicios web.

Algunos ejemplos de empresas que han utilizado PHP en su desarrollo de servicios web son Wikipedia y Wordpress entre muchísimas otras. Vamos a ver por qué, a pesar de la larga existencia de PHP, sigue siendo un lenguaje muy extendido en el mundo profesional.

- Una de las razones por las que PHP caló tanto entre los programadores fue que es un lenguaje relativamente fácil de aprender.
- Como muchos de los lenguajes que hemos visto anteriormente, PHP también es de código abierto. Por tanto, podemos trabajar con el de manera totalmente gratuita.
- El código PHP sólo es interpretado por el lado del servidor en ejecución. Es por eso por lo que su código es invisible a ojos del navegador web y del cliente.
- Debido a que es un lenguaje con mucho recorrido en el sector, PHP ha sufrido cambios muy positivos desde su lanzamiento. Debido a esto, siempre se recomienda a los desarrolladores trabajar con las últimas versiones de PHP ya que cuentan mejoras como por ejemplo la tipificación del código, añadida a partir de PHP 7.

### *Node.js*

Parafraseando a (Patel, no date), Node.js es un lenguaje del lado del servidor que se ejecuta en el motor V8 Javascript (motor que también alimenta a Google Chrome). Este utiliza el modelo de entrada y salida sin bloqueo controlado por eventos, lo que lo hace un perfecto candidato para las aplicaciones con un gran flujo de conexiones.

Esto convierte a Node.js en una opción que, aunque no solventa todos los problemas de los desarrolladores, sí es muy útil en determinadas aplicaciones en las que hay muchas conexiones simultáneas. Es por esto por lo que Node.js se posiciona como un entorno en tiempo de ejecución que aporta mucha escalabilidad y velocidad en aplicaciones muy concurridas por los usuarios.

Node.js funciona con un sólo subproceso, lo que elimina posibles sobrecargas del sistema, ya que no tiene que cambiar entre subprocesos. Este subproceso genera diferentes eventos atendiendo a las solicitudes que le llegan y, en lugar de esperar a que estos eventos se completen para continuar su ejecución, crea una función de devolución que completará tan pronto como pueda o termine otros eventos.

En resumen, estas son algunas de las características más interesantes que nos ofrece Node.js para el desarrollo de aplicaciones de servicio web:

- Como hemos comentado antes, una de sus mayores ventajas es su modelo de peticiones, pues puede atender muchas peticiones simultáneas de usuarios.
- El auge de lenguajes como Javascript ha impulsado mucho el uso de estos entornos es por eso por lo que Node.js cuenta actualmente con mucho soporte por parte de su amplia comunidad de usuarios.
- Cabe recalcar que Node.js no tiene un código realmente complejo. No obstante, puede que tengamos que escribir algunas líneas de más si lo comparamos con entornos como p. ej. PHP.

#### *2.2.2.3. Bases de datos*

A la hora de montar una base de datos para nuestra aplicación, debemos tener en cuenta muchos factores, desde el tipo de base de datos que queremos utilizar siguiendo los intereses de nuestra aplicación, como el servidor que la mantendrá o la herramienta que utilizaremos para la administración de nuestra base de datos.

En la actualidad, la gran mayoría de las aplicaciones, independientemente de si están orientadas a servicios web o a cualquier otro ámbito, cuentan con alguna base de datos que da soporte a toda la información con la que tiene que trabajar la aplicación. Esto se debe a que, con el tiempo, la gestión y el tratamiento de la información ha tomado un papel muy importante en la informática lo que impacta directamente en el volumen de datos que almacena una aplicación.

Con el paso de los años las aplicaciones cada vez almacenan más información de sus usuarios. Si sumamos esto al incremento del uso de la tecnología y, por ende, al aumento de posibles usuarios, tenemos un gran volumen de datos que requieren un proceso muy meticuloso en su tratamiento, ya que realizar modificaciones una vez está construida la aplicación puede ser realmente costoso.

Este apartado pretende ejemplificar todas las decisiones que deberíamos tener en cuenta cuando se está planteando elaborar un sistema de gestión de datos desde cero, desde las decisiones más sencillas hasta las más complejas, viendo en cada nivel que impacto podría tener cada decisión en la elaboración de nuestra base de datos.

En la actualidad, existen muchos tipos de bases de datos según los datos que contengan. Hay bases de datos jerárquicas, de red, transaccionales, relacionales, multidimensionales y orientadas a objetos entre otras. Inicialmente, nos puede parecer abrumador elegir entre tantos tipos de bases de datos. No obstante, la realidad es que, por conveniencia y utilidad, la decisión se suele debatir, en la mayoría de los casos e intereses, entre sólo dos tipos: relacionales o no relacionales.

Antes de determinar si a nuestra aplicación le conviene más una base de datos relacional o una no relacional, tenemos que entender en profundidad qué diferencias tienen, qué nos aportan y de qué carecen. Una buena manera de compararlas y entender sus virtudes o carencias es enfrentarlas en los mismos ámbitos para ver qué nos ofrece cada una.

Las bases de datos relacionales surgen a principios de los años 80 siguiendo el modelo relacional propuesto por Edgar Frank Codd. Este tipo de bases de datos llevan utilizándose de manera profesional prácticamente desde sus inicios, y en la actualidad siguen siendo las más populares por sus propiedades de atomicidad, consistencia, aislamiento y durabilidad, que otorgan a este tipo de bases de datos mucha robustez.

Las bases de datos relacionales son bases de datos en las que su información convive troceada en tablas y estas se relacionan entre sí utilizando identificadores o claves que se encuentran presentes en las tablas. Esto hace que este tipo de bases de datos sean resistentes a vulnerabilidades, sencillas de manejar y fáciles de entender, ya que la información se encuentra siempre estructurada y ordenada.

Por otro lado, tenemos las bases de datos no relacionales. Estas surgieron con posterioridad a las relacionales y no han gozado de popularidad hasta recientemente cuando han ido ganando protagonismo en el sector profesional y se han ido postulando como una alternativa sólida a las bases de datos relacionales.

Como su nombre indica, las bases de datos no relacionales no poseen ninguna clave ni identificador que relacione los datos que almacena. Esto las puede hacer más complejas si las comparamos con las relacionales, ya que no tenemos una estructura precisa sobre lo datos que contiene.

A diferencia de las tablas de las bases de datos relacionales, las no relacionales suelen trabajar con documentos donde se especifica que parámetros contiene y su propio valor. Aunque las bases de datos relacionales trabajen con tablas y las nos relacionales con documentos, tenemos que ser conscientes de que las tablas también se pueden presentar como documentos y viceversa, como vemos en este ejemplo.

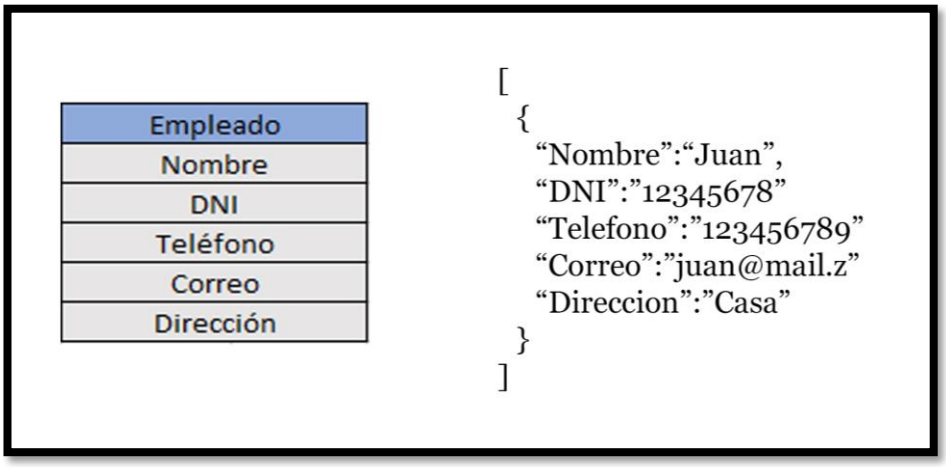



*Figura 2. Formato tabla y formato JSON*

Antes de empezar este apartado, vamos a hacer una breve introducción a SQL, ya que se va a mencionar muchas veces. Parafraseando a (Petković, 2017) SQL recibe su nombre por sus siglas en inglés Structured Query Language o, traducido al español, lenguaje de consultas estructuradas. El origen de SQL está muy ligado con el proyecto llamado System R, el cual fue diseñado e implementado por IBM a principios de los años 80. Este proyecto fue realizado utilizando los fundamentos teóricos del trabajo de E.F.Codd sobre las bases de datos relacionales.

Por otra parte, tenemos JSON o la notación de objeto de Javascript. Se trata de un formato de texto que es muy utilizado tanto en bases relacionales y no relacionales como para la comunicación de todo tipo de aplicaciones. Es interesante decir que JSON surgió como una notación de objeto de Javascript. Sin embargo, ganó tanta popularidad que a partir de 2019 se le reconoce como un formato de texto independiente de Javascript.

Si nos ponemos a investigar sobre las bases relacionales y no relacionales, rápidamente caeremos en la cuenta de que mucha gente hace referencia a ellas como bases de datos SQL o noSQL. Esto puede dar a entender que estas bases de datos utilizan o no SQL. No obstante, cabe destacar que la nomenclatura noSQL hace referencia a not only SQL, es decir, aunque SQL es el lenguaje más utilizado en las bases de datos relacionales y JSON es la notación más utilizada en las bases de datos no relacionales, estos pueden ser intercambiados.

Por tanto, es importante recalcar que SQL puede utilizarse en bases no relacionales y del mismo modo JSON puede ser utilizado en bases de datos no relacionales. De esta manera, podemos trabajar con tablas en una base de datos relacional con nomenclatura SQL y JSON, como también podemos trabajar con SQL y JSON en una base de datos no relacional.

Una vez entendidos los conceptos técnicos de las bases de datos relacionales y no relacionales, podemos pasar a investigar las diferentes herramientas más utilizadas para construir bases de datos. En concreto, en este apartado vamos a realizar un trabajo de investigación sobre los sistemas de gestión de bases de datos o SGBD más utilizados y qué factores pueden ser clave al decantarnos por uno u otro.

Investigando un poco nos encontramos con una página web muy recomendada (solid IT, 2020). Esta página otorga a la mayoría de las bases de datos existentes una puntuación en base a factores clave como pueden ser las menciones que los usuarios hacen de ellas, las ofertas de trabajo que generan o los debates que plantean entre otras.

Por tanto, ya que es imposible hablar de todas las bases de datos del sector, nos parece coherente hablar de las tres bases de datos relacionales y no relacionales con mayor puntuación en db-engines.

Empezando por las bases de datos relacionales, este es el *ranking* de las bases de datos relacionales con la puntuación más alta en el momento de la realización de este trabajo de fin de grado.

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Jun 2020 | May 2020 | Jun 2019 | | | Jun 2020 | May 2020 | Jun 2019 |
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1343.59 | -1.85 | +44.37 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 1277.89 | -4.75 | +54.26 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 1067.31 | -10.99 | -20.45 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 522.99 | +8.19 | +46.36 |
| 5. | 5. | 5. | IBM Db2 | Relational, Multi-model | 161.81 | -0.83 | -10.39 |

*Figura 3. Bases de datos relacionales según puntuación* (DB-Engines, 2020)

Como podemos ver en la imagen, hay una clara diferencia de puntuación entre las tres primeras bases de datos y la cuarta posicionada. Teniendo esto en cuenta, podríamos decir que los tres pilares de las bases de datos relacionales son, en esencia, Oracle, MySQL y SQL Server.

La realidad es que estas tres bases de datos son muy similares en cuanto a características y prestaciones. Los factores que debemos tener en cuenta al elegir son básicamente dos. Antes de seleccionar una debemos conocer el entorno en el que queremos trabajar, pues si nos decantamos por Windows sería recomendable trabajar con Oracle o SQL Server mientras que si vamos a trabajar en Linux es mejor optar por MySQL. Cabe recalcar que estas son recomendaciones de usuarios con experiencia y no significan que, por ejemplo, MySQL no se pueda utilizar en Windows o viceversa.

Por otro lado, otra cosa muy importante que debemos tener en cuenta es el presupuesto con el que contamos, ya que, aunque estas bases de datos tienen su versión gratuita, si necesitásemos una base de datos para un entorno de producción, sería conveniente adquirir una licencia que nos aporte más prestaciones. Este es un breve resumen comparativo sobre las licencias de estas bases de datos y sus precios orientativos asociados. Para utilizar un estándar para todos ellos vamos a suponer cuatro *cores*.

| *Versiones* | *Oracle* | *Microsoft SQL Server (2017)* | *MySQL* |
|---|---|---|---|
| *Standard* | 70.000 $ | 14.868 $ | GRATUITO |
| *Enterprise* | 190.000 $ | 57.024 $ | |

Si nos fijamos en la tabla de precios podemos ver que las licencias de Oracle son mucho más caras que las de SQL Server. Del mismo modo, ambas son muchísimo más costosas que las licencias de MySQL, ya que esta última nos ofrece una base de datos competente de manera gratuita debido a que es de código abierto.

Cabe decir que, aunque no esté reflejado en la tabla anterior, tanto Oracle como SQL Server cuentan con versiones gratuitas con limitaciones que pueden ser utilizables para proyectos personales, pruebas o proyectos de menor complejidad y ambición.

Las principales diferencias entre las versiones Standard y las versiones Enterprise son las funcionalidades extra y herramientas que otorgan un mayor control sobre la base de datos y su disponibilidad.

Por último, vamos a investigar las opciones que existen para trabajar con bases de datos no relacionales. Estas bases de datos están en auge en los sistemas de los últimos años, ya que gozan de ciertas ventajas respecto a las bases de datos relacionales.

Las bases de datos no relacionales son bases de datos descentralizadas y gracias a esta característica son muy escalables. Esto hace que sean bases de datos muy flexibles, ya que pueden ser adaptadas a cambios con mucha más facilidad que una base de datos relacional. Algunos servicios para los que son recomendables las bases de datos no relacionales podrían ser, entre muchos otros, las redes sociales, debido a su gran manejo de volúmenes de datos y la estructura de su información.

Si miramos el *ranking* de db-engines, tenemos que ir hasta la quinta posición para encontrarnos con la primera base de datos no relacional de la lista. Concretamente, estamos hablando de MongoDB.

**MongoDB**

Citando a (Shakuntala Gupta Edward and Navin Sabharwal, 2015), a finales de 2007, Dwight Merriman, Eliot Horowitz y su equipo decidieron desarrollar un servicio online. La intención fue proveer de una plataforma para desarrollar, hosting y para aplicaciones web. Pronto se dieron cuenta de que no existía ninguna plataforma de código abierto que cumpliera estos requisitos.

Por tanto, se decidieron por construir una base de datos que no comulgaba con la tendencia de bases de datos relacionales de la época. Un año después el proyecto estaba listo y preparado para ser utilizado, el servicio no fue lanzado cuando se pretendía y finalmente, en 2009 el equipo desarrollador decidió lanzar MongoDB como una base de datos de código abierto.

En la actualidad, MongoDB es una de las bases de datos no relacionales más populares. Muchas empresas han incluido las características que ofrece MongoDB en sus productos y soluciones. MongoDB utiliza documentos BSON que permiten tener diferentes estructuras, esto implica que la estructura de la base de datos no es fija y va cambiando durante el proyecto.

Cabe remarcar que MongoDB sufrió en 2019 un cambio en su política, publicando así una licencia adicional a las ya existentes y completamente dedicada al uso comercial que contiene más herramientas para los desarrolladores. Esto se debe a que muchos sitios de hosting cobraban por la gestión de MongoDB y el equipo consideró que no era una situación justa explotar de esa manera una aplicación que fue concebida de código abierto y gratuito.

**Apache Cassandra**

La siguiente base de datos no relacional de la que vamos a hablar es Apache Cassandra, que es una base de datos no relacional basada en el modelo clave-valor. Fue desarrollada inicialmente por Facebook y se lanzó en 2008 como un proyecto de código abierto de Google Code. No fue hasta 2010 cuando se postuló como una base de datos competente para la producción.

Apache Cassandra es actualmente una base de datos no relacional de código abierto, lo que implica que puede ser utilizada con fines comerciales y de producción de una manera totalmente gratuita.

**Redis**

La última base de datos no relacional sobre la que vamos a hablar es Redis, una base de datos basada en almacenamiento en tablas de hashes clave-valor. Redis fue inicialmente desarrollada por Salvatore Sanfilippo, quien después fue contratado por VMWare para dedicarse completamente al desarrollo de Redis. En la actualidad RedisLabs posee los derechos de la base de datos Redis.

Redis fue lanzada por 2009 y es una base de datos de código abierto totalmente gratuita preparada para la producción de proyectos comerciales.

### *2.2.2.4. Servidores web HTTP*

Para crear un proyecto de servicio web, será opcional pero recomendable tener algún tipo de software que mantenga nuestra aplicación en internet para que pueda ser accedida o consumida desde una dirección web. Es ahí donde entra el concepto de servidor web. Este tipo de servidores también son conocidos como servidores HTTP, ya que generalmente hacen uso de este conocido protocolo para gestionar todas las conexiones con los usuarios que quieran acceder a nuestro servicio.

En este apartado, vamos a hablar de diferentes tipos de servidores web, todos orientados al mismo propósito mencionado anteriormente. El primer servidor del que vamos a hablar es también el más antiguo y probablemente más utilizado, Apache. Seguidamente, hablaremos de Nginx, LiteSpeed y por último de un servidor desarrollado por el gigante de la tecnología, Microsoft.

Consideramos interesante que, antes de meternos en materia, echemos un vistazo a esta gráfica realizada por W3 Techs. Esta página web se dedica a recopilar información sobre la utilización de determinadas herramientas tecnológicas con el objetivo de crear documentos que nos ayuden a entender un poco más la tecnología. Según W3 Techs estos son los servidores web más utilizados a 22 de junio de 2020.

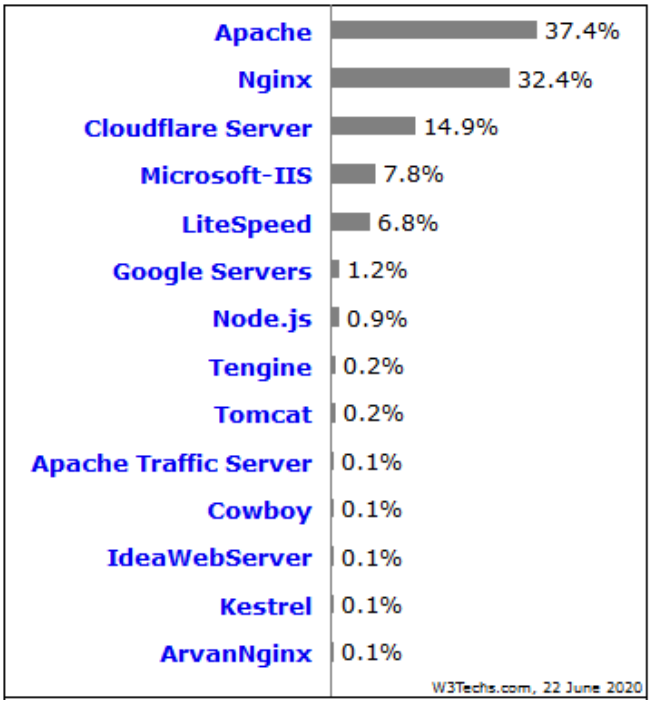



*Figura 4. Servidores web más utilizados internacionalmente* (W3Techs, 2020)

Como podemos ver, Apache y Nginx se llevan casi todo el trozo del pastel de los servicios web. Esto significa que aproximadamente el 70% de los servicios web que visitamos y utilizamos se encuentran bajo servicios de Apache o Nginx. Por otro lado, vemos como los servidores web de Microsoft y LiteSpeed quedan muy por detrás de estos. Sin embargo, nos parece muy interesante hablar de ellos debido a las empresas que los gestionan y sus características.

Es importante mencionar que la mayoría de las herramientas que van a nombrarse en este apartado son capaces de realizar los servicios básicos de la comunicación HTTP y no tiene mucho impacto la herramienta que elijamos para nuestro proyecto en cuestión. No obstante, siempre es interesante conocer un poco más sobre estas herramientas, las diferentes opciones, su historia y las empresas que se encuentran detrás de ellas.

### Apache

Según la propia documentación de Apache (Apache, 1995), el proyecto de servidor Apache HTTP es un esfuerzo por conseguir desarrollar y mantener un servidor HTTP de código abierto para sistemas operativos modernos como Windows y UNIX. El objetivo del proyecto es ofrecer una herramienta segura, eficiente y extensible que brinde servicios HTTP acordes a los estándares actuales HTTP.

Apache fue publicado en 1995 y ha sido el servidor web más popular desde abril de 1996 y no ha dejado de serlo desde entonces. Recientemente han publicado su versión 2.4 y han celebrado su 25 aniversario.

### Nginx

Recurriendo a la propia documentación de Nginx (Nginx, 2004), Nginx es un servidor HTTP, proxy inverso HTTP, servidor proxy de correo y servidor genérico para los protocolos TCP/UDP. Nginx fue originalmente escrito por Igor Sysoev para gestionar grandes tráficos en la red utilizando un servidor ligero y sencillo.

Fue publicado en 2004 bajo una licencia abierta, aunque cabe destacar la existencia de una licencia de pago conocida como Nginx plus. En sus inicios Nginx fue utilizado mayoritariamente en Rusia, dónde fue desarrollado, aunque, debido a sus características, rápidamente se extendió por todo el mundo hasta ser el segundo servidor web más utilizado actualmente.

### LiteSpeed

Citando un artículo de (*PR Newswire*, 2007), el desarrollo del servidor web de LiteSpeed comenzó en 2002 aunque la primera versión fue lanzada en 2003. El servidor web de LiteSpeed se postula como una alternativa a Apache ya que tienen la misma configuración y gozan de bastante compatibilidad entre ellos.

LiteSpeed se caracteriza por mejorar la capacidad para gestionar miles de conexiones concurrentes, un rendimiento sobresaliente, mucha seguridad, actualizaciones en línea automáticas y una interfaz de usuario sencilla para la administración. En los últimos años LiteSpeed ha ido creciendo hasta casi superar a la herramienta de Microsoft para servicios web. Actualmente, se utiliza aproximadamente en el 7% de todas las páginas web de internet.

### Microsoft Internet Information Services (IIS)

Basándonos en el libro Microsoft IIS 10.0 Cookbook (Khan, no date), Microsoft Internet Information Services más conocido como IIS es un servidor web desarrollado por Microsoft. Este fue introducido junto a Windows Server 2016 y Windows 10.

Microsoft ofrece características HTTP, HTTPS, SMTP, SNMP, FTP, FTPS lo que lo convierte en uno de los servidores web más completos. Sin embargo, el éxito de IIS como servidor web reside más en la gran seguridad que ofrece respecto a otros servidores web como Apache y su manera de gestionar el tráfico para operar con muchas conexiones.

Otra de las cosas que más nos ha gustado de este servidor web es su facilidad para estar integrado con el sistema operativo Windows. Si utilizamos una versión de Windows Server, sólo bastará con activar la característica, mientras que en una versión de Windows estándar tendremos que descargarlo. Una vez descargado, podemos alojar una página web en internet en pocos minutos, ya que cuenta con una interfaz muy fácil de entender.

## 2.3. Servidor

Hasta este punto, todas las herramientas que hemos visto para la creación de aplicaciones web son excelentes. No obstante, estas aplicaciones web necesitan de un entorno donde ser ejecutadas para que puedan funcionar. Es en este momento en el que entra la elección de entorno para el funcionamiento de nuestra aplicación web.

Para elegir bien qué servidor necesitamos para ejecutar nuestra aplicación tenemos que entender bien qué queremos ofrecer y con qué calidad. Si nuestra aplicación va a soportar mucho tráfico, debemos elegir bien los componentes de nuestro servidor, ya que una mala elección podría arruinar por completo la experiencia de nuestra aplicación y en consecuencia ser un fracaso, por muy bien desarrollada que esté.

Para la elección de nuestro servidor deberemos tener en cuenta diferentes factores. Estos son los que vamos a valorar para la selección de nuestro servidor y que pueden ser aplicables a cualquier tipo de proyecto:

- *Presupuesto:* es muy importante entender las limitaciones económicas que tiene nuestro proyecto para así elegir el servidor que más se adecua a estas limitaciones.
- *Escalabilidad:* deberemos ser capaces de determinar si nuestra aplicación puede sufrir grandes cambios en el volumen de tráfico para adquirir un servidor con mejores prestaciones o para decantarnos por otro más modesto.
- *Disponibilidad:* puede que, por el carácter natural de nuestra aplicación, esta deba tener una alta disponibilidad y por ello tengamos que optar por servicios en la nube o servidores con fuentes de energía alternativas en caso de fallo.
- *Prestaciones:* debemos tener en cuenta en todo momento los requerimientos de prestaciones que necesita nuestra aplicación. Por ejemplo, si nuestra aplicación realiza muchas consultas a una base de datos, es probable que nos interese aumentar la memoria RAM para mejorar la velocidad y la respuesta de estas consultas.
- *Espacio:* aunque tengamos mucho presupuesto y tengamos claro qué prestaciones necesitamos, siempre hay que tener en cuenta el espacio como un limitador físico. En la actualidad existen muchos tipos de servidores, algunos más compactos y otros que ocupan más espacio. Si nuestro espacio tangible es limitado, esta decisión puede tener consecuencias en la refrigeración del servidor y podría repercutir directamente en el rendimiento.
- *Entorno:* conocer el sistema operativo y las herramientas con las que queremos trabajar es indispensable, pues estos factores pueden determinar tanto el precio del servidor, como su rendimiento.

A continuación, vamos a investigar qué tipos de servidor elegir teniendo en cuenta todas las características expuestas anteriormente. En este caso hemos decidido dividirlos esencialmente en tres tipos distintos de servidores: los servidores o máquinas físicas, los servidores virtuales o máquinas que utilizan virtualización dentro de equipos físicos y, por último, hablaremos de servidores en la nube.

### 2.3.1. Servidor físico

Los servidores físicos son probablemente el concepto de servidor más sencillo y extensamente utilizado en el mundo de la informática. Este tipo de servidores son utilizados tanto en pequeños proyectos como en otros más exigentes. Sin embargo, estos son más comunes en proyectos más pequeños o con menos exigencias.

En la actualidad, existen muchos tipos de servidores físicos que se pueden adaptar perfectamente a cualquier tipo de necesidades que nos puedan surgir en nuestros proyectos. De entre todos estos servidores, los tres más utilizados y en los que vamos a hacer hincapié en este apartado son principalmente los servidores torre, servidores tipo rack y, para finalizar, los servidores tipo blade.

Cada tipo de servidor mencionado en el anterior párrafo tiene diferentes costes, funcionalidades, prestaciones, espacio, etc. Es por este motivo por el que vamos a realizar un estudio de cada tipo de servidor físico explicando todo lo que nos puede ofrecer, prestando atención a las características descritas precedentemente.

**Servidores torre**

Los servidores tipo torre son la expresión más sencilla de un servidor. Este tipo guarda algunas similitudes con los ordenadores personales de sobremesa que puede utilizar cualquier usuario. No obstante, la mayor diferencia entre ellos son las prestaciones.

Los servidores de tipo torre suelen estar equipados con prestaciones muy superiores a las de los ordenadores de sobremesa que tenemos en nuestras casas y, aunque por su exterior pueda parecer que no haya diferencia alguna, por su interior son muy distintos.

Por lo general, los servidores torre tienen mejoras significativas en la memoria principal o RAM. Esta prestación permite al procesador guardar más información de sus cálculos y, por consecuente, trabajar con más procesos al mismo tiempo. Por otro lado, otra característica que suele diferenciar un servidor de un ordenador convencional suele ser el procesador. Normalmente los servidores cuentan con procesadores que permiten realizar cálculos mucho más rápidos y complejos que los convencionales, lo que impacta directamente en el rendimiento.

Además, la refrigeración de estos equipos es diferente a la de los ordenadores convencionales, ya que cuentan con piezas más potentes. Estas sufren más de recalentamiento y por tanto necesitan un buen sistema de refrigeración para su adecuado funcionamiento. Es por esto por lo que los servidores cuentan con ventiladores especiales que provocan diferentes flujos de aire dentro de la máquina, con el único objetivo de mantener una temperatura óptima para exprimir al máximo sus prestaciones y protegerlos de sobrecalentamientos.

### Servidores tipo rack

Los armarios racks son estructuras metálicas con diferentes soportes también metálicos, como si de una estantería se tratase, en la que podemos colocar distintos elementos orientados a la electrónica, informática o comunicaciones. La anchura de estos estantes está normalizada en todo el sector para que podamos instalar cualquier dispositivo independientemente del fabricante.

Estas medidas son conocidas como unidades rack (U), que equivalen a 44.45 mm de altura. De esta manera, un servidor tipo rack puede ocupar un número entero de unidades rack. Por otra parte, la anchura del rack suele estar en el estándar de los 60 cm y 80 cm de profundidad.

*Figura 7. Ejemplo de armario para colocar servidores tipo rack* (QLOUDEA, 2018)

Las principales ventajas que ofrecen este tipo de servidores son esencialmente la considerable reducción y aprovechamiento del espacio debido al tamaño que tienen y a que son apilables. Por otro lado, este formato modular facilita también en gran medida la instalación y ordenación de cables.

### Servidores blade o cuchilla

Los servidores tipo blade o cuchilla son servidores que tienen procesador, memoria principal, algunas tarjetas y en determinados casos discos duros. Este tipo de servidor no puede funcionar por sí mismo, ya que no tiene fuente de alimentación, ni refrigeración, ni siquiera tarjeta de red.

Este tipo de servidores se apilan en chasis que posteriormente se acoplan a un armario rack. Este chasis suele tener una altura de alrededor de 4 a 6 U y cada chasis puede albergar hasta dieciséis servidores de tipo cuchilla. Es el chasis el dispositivo que lleva incluido la fuente de alimentación, ventilación, tarjeta de red y todo tipo de cableado.

Los recursos de los chasis mencionados anteriormente son compartidos por todos los servidores. Esto resulta en una reducción del coste, ya que no necesitamos electrónica para cada servidor, una significante reducción del espacio que se utiliza. Por otro lado, este tipo de servidores es mucho más versátil que los vistos anteriormente, pues se pueden intercambiar servidores sin detener el sistema y utilizando tecnología de intercambio en caliente.

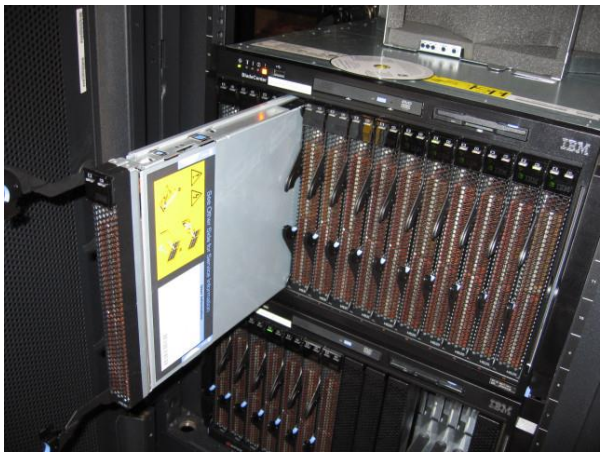



*Figura 5. Armario para servidores tipo cuchilla o blade* (TicoStyle, 2009)

Esta tecnología los hace muy sencillos de gestionar ya que pueden ser sustituidos sin necesidad de tocar ningún tipo de cable o instalación, si tenemos en cuenta esto y que algunos de ellos no tienen un disco físico que pueda fallar, obtenemos un sistema muchos más robusto, sencillo y menos propenso a fallos.

### 2.3.2. Servidor virtual

Otra opción que valorar si necesitamos un servidor podría ser la virtualización de este en una máquina, pero primero tenemos que entender qué es virtualizar un equipo, qué supone en nuestros proyectos, qué ventajas nos puede generar, así como qué factores debemos tener en cuenta.

En esencia, la virtualización de un servidor en un ordenador es almacenar otro sistema en una partición de la propia máquina física que estemos utilizando. De esta manera, podemos asignar ciertos recursos de un servidor físico y destinarlo a distintos propósitos. Es decir, podemos utilizar un servidor físico que adquiramos para diversos fines virtualizando y dividiendo sus recursos en sistemas distintos con distintas finalidades.

Estas virtualizaciones son más conocidas como servidor virtual privado o por su abreviatura SVP. También son muy conocidos por su nombre en inglés, *virtual private server,* o su abreviatura, VPS. En este caso, nos referiremos a ellos como SVP.

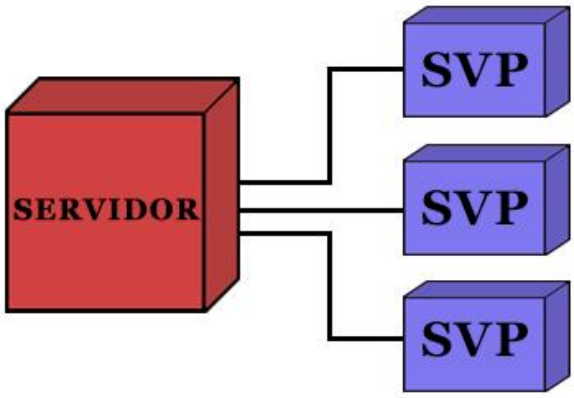



*Figura 6. Ejemplo visual de un particionado de servidores virtuales privados*

Como se puede ver en la imagen, un servidor puede suministrar diferentes tipos de sistemas tales como servidores de correo, servidores de dominio, servidores para servicios en línea o aplicaciones web entre otras.

Estos servidores suelen ser normalmente compartidos, aunque también existen dedicados. Es decir, si tenemos en cuenta que cada servidor virtual privado es autónomo e independiente y puede contener muchos tipos de proyectos, lo habitual es que si nos decantamos por alguno de estos servicios estemos compartiendo recursos físicos con proyectos y usuarios distintos.

En muchos casos, cuándo un proyecto está en una fase inicial, un servidor virtual privado con recursos o servidor compartidos es más que suficiente para satisfacer cualquier necesidad. No obstante, conforme los proyectos van creciendo, muchas empresas acaban contratando estos servicios en grandes empresas con prestaciones que se adaptan mejor a sus requerimientos.

Por tanto, se recomienda hacer uso de este tipo de servidor para proyectos pequeños, proyectos en fase inicial, realizar pruebas personales o desarrollos del servicio, aplicación o proyecto que se quiera realizar.

Estas son, a grandes rasgos algunas de las ventajas que nos proporciona utilizar servidores virtuales privados:

- *Escalabilidad:* cuando utilizamos un servidor virtual, en caso de que nuestro proyecto requiera más recursos, simplemente tendremos que asignarlos si la máquina es nuestra o contratarlos si nos lo proporciona una empresa.
- *Limitaciones:* por otro lado, debemos tener presente que este tipo de servidores tiene una limitación y es la de su propia máquina física. Entre todos los VPS que haya en una máquina, nunca puede superar los recursos totales del servidor físico.
- *Económico:* como somos nosotros quienes decidimos los recursos que tenemos, podemos ajustar el precio exactamente a los recursos que vayamos a necesitar y pagar sólo por lo que necesitemos.
- *Eficiencia y seguridad:* los servidores virtuales se pueden guardar en un archivo que puede ser rápidamente migrado a otra máquina en caso de emergencia.
- *Variedad:* existen muchos tipos de software para virtualizar sistemas, algunos de pago y otros no. Algunos ejemplos son VirtualBox, VMware, QEMU o HyperV entre otros.

### 2.3.3. Servidor en la nube

En el apartado anterior hemos visto a grandes rasgos qué es un servidor compartido y en qué casos se utiliza. Sin embargo, tenemos que entender también el concepto antagónico: servidores dedicados. Los servidores dedicados son aquellos en los que disponemos de todos los recursos de la máquina física, es decir, que no compartimos los recursos con nadie más que con nuestro proyecto.

Una vez entendido este concepto ya podemos ahondar más en qué consisten los servidores en la nube, qué nos aportan y algunos ejemplos de empresas que trabajan con servidores en la nube para sus proyectos entre otros conceptos.

Los servidores en la nube guardan muchas similitudes con la virtualización de servidores que hemos explicado con detalle en el punto anterior. No obstante, sus diferencias son su mayor y más interesante atractivo. De la misma manera que en la virtualización de servidores, para crear servidores en la nube también se utilizan las mismas herramientas de virtualización para fragmentar servidores físicos en distintos servidores virtuales a los que acceder remotamente.

En este caso, el usuario que contrata un servicio de hosting tiene la percepción de que trabaja en un servidor dedicado. Sin embargo, la gran ventaja de este tipo de servicios es en concreto la abstracción del hardware.

La abstracción del hardware significa que el hardware que utilicemos en un servidor en la nube no se encuentra realmente en ningún sitio, es decir, que nuestro almacenamiento o la red no tiene una ubicación predeterminada fija. Esto nos ofrece muchas ventajas en el caso de que hubiese algún problema, ya que, si fallara algún nodo de la empresa que tenemos contratada, se nos redireccionaría a otro y seguiríamos funcionando con total normalidad.

De este modo, una nube tiene distintos nodos que están funcionando y dando soporte a multitud de proyectos. Si unimos esta característica a la escalabilidad de los servidores virtuales privados, hacen de estos la mejor opción para ofrecer un servicio en línea con mucha disponibilidad y eficiencia.

Es por estos motivos que la tendencia en la actualidad es contratar a servicios en la nube dónde poder virtualizar nuestro proyecto a prueba de errores. Empresas de países como EE. UU. o China cuenta con la mayoría de sus servicios virtualizados en la nube, mientras que en Europa los datos dicen que aún nos queda camino por recorrer.

## 2.4. Seguridad y ámbito de red

La seguridad tiene un papel protagonista independientemente del proyecto que estemos desarrollando. Es un concepto en el que las organizaciones se fijan cada vez más, ya que de él dependen aspectos tan importantes como la confianza del cliente en nuestros servicios. Es por esto por lo que siempre que desarrollamos un proyecto en línea tenemos que estudiar bien desde dónde y cómo va a ser accesible.

En este punto podemos empezar por determinar el ámbito de red de nuestra aplicación, es decir, si nuestro servicio va a poder ser accesible desde internet por cualquier tipo de usuario o más bien es un tipo de servicio que se va a utilizar sólo en redes locales que sólo utilizan intranet para la comunicación interna.

En el caso de que nuestro servicio no vaya a tener salida a internet y sólo vaya a comunicarse mediante intranet, tan sólo tendremos que tomar medidas habituales de protección sobre cualquier red local. Muchas de las debilidades de estos sistemas son más humanas que físicas. Es por eso por lo que una buena medida podría ser poner normativas o protocolos que impidan a los usuarios de nuestra red la conexión a dispositivos que estén fuera de la organización y así evitar una posible intrusión no deseada.

Si nuestro servicio tiene que ser accesible desde internet, las cosas se ponen un poco más complejas y deberemos tomar más precauciones, ya que esto implica que cualquier usuario desde cualquier parte del mundo va a tener la oportunidad de acceder a nuestro servicio y, en el caso de que tenga alguna debilidad, explotarla. Vamos a presentar algunas técnicas o herramientas que podemos aplicar para mejorar la protección de nuestra red local si la vamos a convertir en accesible.

### Cortafuegos o firewall

Primeramente, deberíamos considerar colocar diferentes cortafuegos en nuestra red. Un cortafuego es, en esencia, un dispositivo que se encarga de filtrar el tráfico de la red a la que se conecte. A estos dispositivos se les pueden configurar reglas específicas de tráfico como por ejemplo que sólo acepte tráfico en puertos determinados o de algunas direcciones determinadas. Existen muchas configuraciones de cortafuegos. Esta es una de las más sencillas, utilizando únicamente un cortafuegos que filtra el tráfico para toda nuestra red local.

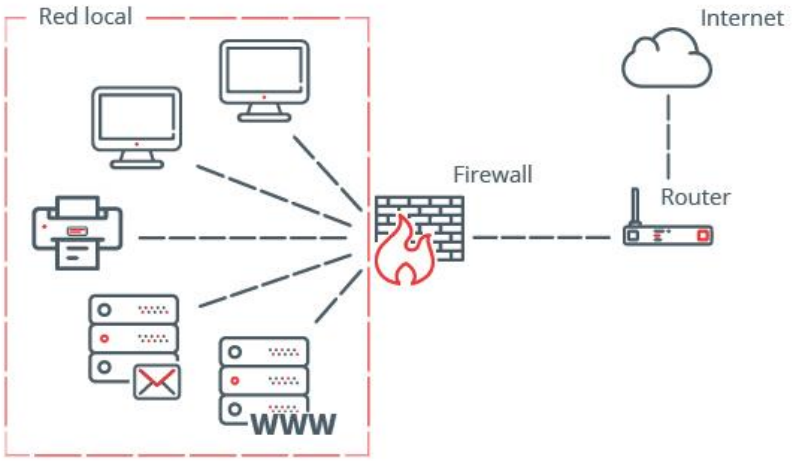



*Figura 7. Configuración de una red utilizando un cortafuegos* (INCIBE, 2019)

### Zonas desmilitarizadas o DMZ

Si se quisiese aumentar la seguridad, podríamos instalar una zona desmilitarizada o DMZ. Se trata de una red aislada que se encuentra dentro de la red interna de la organización. En ella se ubican exclusivamente todos los recursos de la empresa que deben ser accesibles por internet, como servidores web o de correo. Este es un ejemplo de zona desmilitarizada.

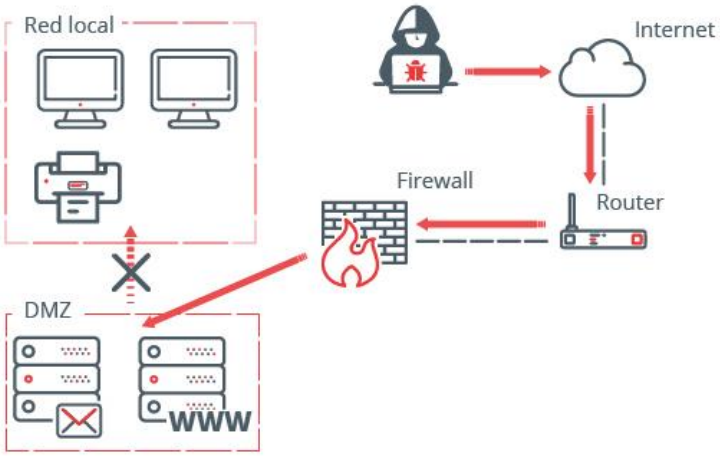



*Figura 8. Configuración de red utilizando DMZ* (INCIBE, 2019)

Tal y como está representado en la ilustración anterior y por lo general, una DMZ permite las conexiones procedentes tanto de Internet, como de la red local de la empresa donde están los equipos de los trabajadores, pero las conexiones que van desde la DMZ a la red local no están permitidas. Esto se debe a que los servidores accesibles desde internet son más susceptibles a sufrir un ataque que pueda comprometer su seguridad. Si un ciberdelincuente comprometiera un servidor de la zona desmilitarizada, tendría mucho más complicado acceder a la red local de la organización, ya que las conexiones procedentes de la DMZ se encuentran bloqueadas.

Si se quisiera aumentar aún más la seguridad en la organización, se podría utilizar otro cortafuego intercalado entre el primer cortafuego del DMZ con nuestra red local como se puede ver en la siguiente imagen. (INCIBE, 2019)

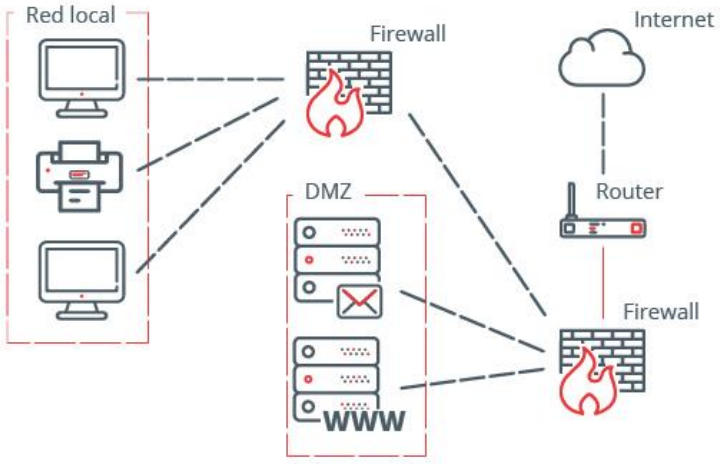



*Figura 9. Configuración de red utilizando dos cortafuegos* (INCIBE, 2019)

## 2.5. Estrategias de almacenaje

En la actualidad, existen diferentes estrategias de almacenaje en los almacenes según el tipo de producto que se quiera almacenar o sus condiciones especiales, no obstante, todas ellas se pueden agrupar en sistemas de posición fija o sistemas de posición aleatoria.

En los sistemas de posición fija, cada producto ocupa siempre una posición permanente dentro del almacén, por lo cual existe una relación biunívoca entre hueco disponible y producto almacenado, de tal manera que cuando no hay stock, el huevo queda vacío pero reservado para el producto asignado; mientras que en los sistemas de posición aleatoria o caóticos, los productos se ubican en cualquier huevo que esté vacío dentro del almacén, pudiendo cambiar la posición del mismo en función del espacio disponible y criterios de productividad. (Tejero, 2008)

Como suele ocurrir en estos casos, cada sistema tienes sus ventajas e inconvenientes y la elección entre uno de ellos dependerá en gran medida de las características del almacén que se quiere construir. Es por este motivo por el que para elegir entre una de las dos estrategias deberemos realizar con anterioridad un estudio sobre la organización y los requisitos de esta.

**Ventajas e inconvenientes de sistemas con posiciones fijas**

Estas son algunas ventajas de utilizar la estrategia de posiciones fijas:

- Este tipo de almacenamiento es muy útil para casos en el que el sistema no se encuentra informatizado y los empleados operan de forma manual el almacén ya que, las posiciones fijas ayudan en la identificación y la organización de los productos.
- Por otra parte, en este tipo de almacenes los productos que se reciben ya tienen asignadas una zona o posición específica, esto facilita la ubicación manual de los productos en el almacén ya que los operarios saben en todo momento dónde ubicar cada producto

Estas las principales desventajas de utilizar una estrategia de posiciones fijas:

- Para este tipo de almacenaje se necesita mucho espacio, esto es debido a que, si tenemos gran parte del almacén dedicado a un tipo de producto y nos quedamos sin espacio para otros tipos de productos, no podremos utilizar las ubicaciones libres asignadas a otros productos.
- Si como consecuencia del punto anterior necesitamos ampliar el almacén, el precio se va a disparar y es probable que estemos desperdiciando mucho espacio libre del almacén.
- Teniendo en cuenta los puntos anteriores, si nuestro almacén crece con el paso del tiempo, llegaremos a un punto en el que será muy complicado e ineficiente realizar una correcta gestión de estos almacenes.

**Ventajas e inconvenientes de sistemas con posiciones aleatorias**

Estas son las principales ventajas que podemos encontrar al utilizar almacenes de posiciones aleatorias:

- Este tipo de sistemas es inconcebible para sistemas manuales, pero altamente recomendable para sistemas informatizados, esto se debe a que un operario no es capaz de recordar ni calcular la ruta óptima para extraer los productos del almacén, pero un ordenador puede mantener y utilizar toda esta información sin problemas.
- Se reduce considerablemente el tamaño del almacén necesario para operar ya que se puede utilizar cualquier posición para cualquier producto.
- Respecto a la estrategia de posiciones fijas, las posiciones aleatorias reducirían drásticamente el coste de un almacén y simplificaría el trabajo de los operarios que simplemente tendrían que seguir las indicaciones del SGA que gestione el almacén.
- Podemos saber en todo momento el estado del almacén ya que el sistema informático debe ser capaz de ofrecernos en todo momento una visión general del almacén.

Esta son algunas de las desventajas de utilizar sistemas de posiciones aleatorias en almacenes:

- Este tipo de estrategias requieren de realizar una fuerte inversión en el desarrollo de una aplicación para la gestión del almacén, como se ha comentado anteriormente realizar estas tareas de manera manual sería realmente ineficiente.

# 3. Análisis del problema

## 3.1. Introducción

Ahora que ya somos conocedores de muchas de las herramientas y el contexto que se emplea en la actualidad para desarrollar aplicaciones y servicios para la gestión de almacenes, es el momento de realizar nuestra primera toma de contacto con el problema. En este apartado realizaremos reuniones con los clientes para que nos transmitan el problema o el principal motivo por el cual se decantan por una solución informática.

Las especificaciones iniciales de una aplicación tienen que ser muy sencillas y superficiales, ya que debemos de tener siempre en mente que probablemente los interesados tienen unos conocimientos limitados de informática y en consecuencia vamos a recibir una visión muy poco técnica del problema.

Es por eso por lo que en este apartado nuestro objetivo tiene que ser identificarnos al máximo con el cliente para localizar de la manera más precisa posible el problema que tiene y ofrecer la solución más ajustada a sus necesidades.

En algunas ocasiones se subestima el potencial del análisis del problema y nuestra tendencia como técnicos es especificar directamente utilizando detalles demasiado especializados y complejos. Esto puede ocasionar que el resultado de la aplicación termine siendo de una utilidad que difiere completamente con el propósito con el que la concibió el cliente.

En nuestra opinión, es importante utilizar el tiempo que sea necesario. Un buen análisis e identificación del problema resultarán en la mayoría de los casos en una aplicación mucho más adaptada y flexible a las necesidades de nuestro cliente y por consecuente, obtendremos un mayor grado de satisfacción.

A propósito de documentar de manera adecuada esta etapa de análisis, se considera que es una buena opción basarnos en la ingeniería de requisitos como medio para la llevarla a cabo. En concreto, se ha decidido basar tanto la estructura como el contenido de este punto en la investigación realizada por los doctores Amador Durán Toro y Beatriz Bernárdez Jiménez en el campo de elicitación de requisitos de sistemas software. (Durán Toro and Bernárdez Jiménez, 2002).

## 3.2. Participantes en el proyecto

En este punto se va a realizar en primer lugar una identificación de todos los participantes de este proyecto, una vez identificados vamos a intentar elaborar unos perfiles sobre los participantes con el único objetivo de conocerlos mejor y empatizar más con ellos. Para simplificar este apartado vamos a dividir los participantes del proyecto en clientes y desarrolladores.

### 3.2.1. Cliente

El primer paso para empatizar con el problema es ser conocedores de la trayectoria de la empresa. Para ello vamos a responder un conjunto de preguntas habiendo indagado sobre la empresa en su propia página web.

**¿Cómo se definen?**

Disoltec es una empresa dedicada a los servicios integrados de tecnología formada por un equipo de profesionales altamente cualificados y con gran experiencia en la búsqueda e implantación de soluciones tecnológicas, aportamos a las empresas un valor añadido a la actividad que desempeñan.

El trato directo y personalizado, así como el conocimiento de las necesidades de sus clientes son algunas de las principales ventajas competitivas de Disoltec. (Disoltec, 2007a)

**¿Qué ofrecen?**

Disoltec desarrolla soluciones que son algo más que un simple programa o unos elementos de hardware. Con el concepto de solución queremos englobar todo tipo de trabajo que somos capaces de realizar llave en mano para nuestros clientes, de forma que somos capaces de resolver el problema que puedan tener o conseguir la mejora de sus procesos de producción y gestión.

De ahí el amplio abanico de soluciones que ofrecemos, desde webs o infraestructura de comunicaciones hasta desarrollo software, pasando por productos propios o servicios de mejora de los procesos empresariales. Nuestras soluciones muchas veces combinan más de uno de estos campos de actividad. (Disoltec, 2007b)

**¿Cuánta experiencia tienen?**

Disoltec lleva más de 13 años ofreciendo soluciones tecnológicas profesionales en ambientes de producción, trazabilidad y estocaje de cualquier tipo de producto, aunque en la actualidad están más enfocados a la automoción. Sus soluciones y profesionalidad han colocado a la empresa tecnológica a trabajar con algunas de las empresas más importantes de la automoción.

Algunos de sus clientes son Indra, Acciona, Gamesa, Walkerpack, Modular Logística Valenciana, Fabricación Modular Valenciana, Integra y Ford, entre otros.

### 3.2.2. Desarrolladores

En este punto se debería realizar una investigación similar a la expuesta en el apartado anterior, analizando la trayectoria del equipo de desarrolladores, experiencias, clientes previos, integrantes e incluso sería interesante recalcar el impacto que tienen sus aptitudes para el proyecto o qué partes específicas del proyecto van a desarrollar cada uno.

Por tanto, hay que tener en mente que, habitualmente los desarrolladores de las aplicaciones suelen ser un equipo de técnicos con experiencia que realizan muchas tareas entre todos para alcanzar el mismo objetivo: crear una aplicación lo más profesional posible y que se ajuste al máximo al problema que el cliente quiere solventar.

No obstante, debido al contexto excepcional que supone la realización de un trabajo de fin de grado, este trabajo será enteramente desarrollado por el alumno basándose en la opinión de profesionales que conocen de primera mano el sector y pueden ayudar a despejar dudas que de por sí solo al alumno le serían complejas de resolver.

## 3.3. Descripción del sistema actual

Para entender correctamente el funcionamiento actual del sistema con el que trabaja la empresa, se celebró una primera reunión entre algunos de los empleados de Disoltec y el autor de este trabajo de fin de grado. El único objetivo de esta reunión fue documentar el método de trabajo, entender las limitaciones y problemas del sistema actual y empatizar con el cliente.

| | |
|---|---|
| *Fecha reunión* | 19 de junio de 2020 |
| *Cliente* | Diseño de soluciones tecnológicas |
| *Proyecto* | Software para la gestión de los almacenes de Disoltec |
| *Razón* | Describir todos los protocolos, mecánicas y procedimientos del sistema actual |

*Descripción del sistema actual*

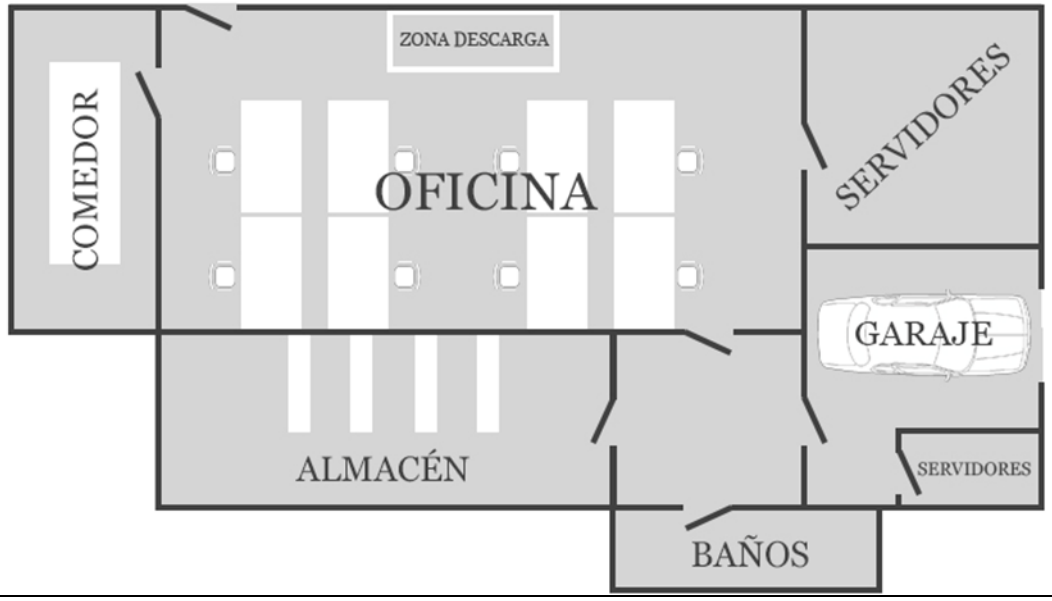
Disoltec es una empresa que, debido a las soluciones tecnológicas que ofrecen, en muchas ocasiones distribuye todo tipo de componentes tecnológicos de diferentes tamaños a sus clientes. Desde elementos muy pequeños como pueden ser cables a bultos más grandes como podrían ser servidores.

En su funcionamiento actual, el modo de proceder cuando reciben estos paquetes suele ser variado. Si se trata de un producto que puede ser utilizado próximamente, puede permanecer hasta varios días en el espacio donde se recibe hasta su salida de las oficinas. Por otra parte, existen ocasiones en la que estos productos se compran para tener repuestos para posibles instalaciones y su salida de las oficinas es indefinida.

Una vez reciben estos productos, normalmente por los empleados, no se registra su entrada en los almacenes, ni su salida. No obstante, se nos hace saber que sí se registra su compra en un software de gestión económica.

Aparte de esto, los empleados nos transmiten que también mantienen los sistemas de sus clientes. Esto hace que normalmente reciban dispositivos disfuncionales que, o bien deben ser reparados por ellos mismos en sus oficinas, o tienen que ser enviados a alguna empresa especializada en su tratamiento.

Los propios empleados nos enseñan las instalaciones y se aprecia que tienen dentro de las oficinas un almacén con estanterías de diversos tamaños y, por otro lado, un aparcamiento que también tiene estantes en los que guardan diferentes tipos de productos.

| *Documentos adjuntos* |
|---|
| Ya que los empleados no tienen ningún esquema de las instalaciones a mano, decidimos que es interesante realizar un bocetado de las instalaciones para entender un poco mejor el funcionamiento del sistema actual.<br>  <br> |

# 4. Especificación de los requisitos

## 4.1. Objetivos del sistema

Para especificar los objetivos del sistema se concertó una segunda reunión con Disoltec. En esta reunión se espera que se nos desarrolle qué desean conseguir con la aplicación, es decir, cómo quieren gestionar su almacén y qué acciones les gustaría realizar.

| **OBJ-01** | **Gestionar las entradas de productos** |
|---|---|
| **Descripción** | *El sistema deberá gestionar toda la información correspondiente a los productos que entran en el almacén y permitir procesarlos* |
| **Estabilidad** | Alta |
| **Comentarios** | Ninguno |

| **OBJ-02** | **Gestionar las salidas de los productos** |
|---|---|
| **Descripción** | *El sistema deberá gestionar toda la información correspondiente a los productos que salen del almacén y llevar un registro correcto.* |
| **Estabilidad** | Alta |
| **Comentarios** | Ninguno |

| **OBJ-03** | **Trazabilidad del producto** |
|---|---|
| **Descripción** | *El sistema deberá ser capaz de mostrar en todo momento la situación de cada producto, independientemente de en qué etapa se encuentre.* |
| **Estabilidad** | Alta |
| **Comentarios** | Ninguno |

| **OBJ-04** | **Gestión del stock** |
|---|---|
| **Descripción** | *El sistema deberá conocer en todo momento la cantidad de productos que le quedan de cada categoría y avisar a los operarios en caso de que fuese recomendable realizar una nueva compra para tener stock.* |
| **Estabilidad** | Alta |
| **Comentarios** | Ninguno |

| **OBJ-05** | **Gestionar partes de trabajo** |
|---|---|
| **Descripción** | *El sistema debe ser capaz de gestionar los partes de trabajo que los operarios le insertan, estos partes se realizan con el objetivo de tener un registro de las horas realizadas a cada cliente.* |
| **Estabilidad** | Alta |
| **Comentarios** | Ninguno |

## 4.2. Requisitos del sistema

### 4.2.1. Requisitos funcionales

Una vez identificados los requisitos de información, es el momento de trabajar los requisitos funcionales de nuestro sistema. Sin entrar mucho en detalle, los requisitos funcionales son aquellos que detallan el funcionamiento de nuestro sistema. Es decir, en este apartado se define de manera detallada paso por paso las entradas, procesos y salidas que se espera que nuestro sistema sea capaz de realizar.

Como se ha comentado con anterioridad, es muy importante que este proceso sea realizado juntamente con nuestro cliente. Rellenaremos de esta manera ciertos documentos relacionados con los requisitos funcionales que nos ayudarán a paliar tanto la falta de propuestas como a simplificar la comunicación entre los integrantes del proyecto.

Para pautar este cometido vamos a intentar dividir primeramente nuestro proyecto en subproyectos con acciones más sencillas. Esta simple tarea nos va a permitir ser mucho más concisos con los requisitos y evitar distraernos con otras funcionalidades que puedan empañar la correcta especificación.

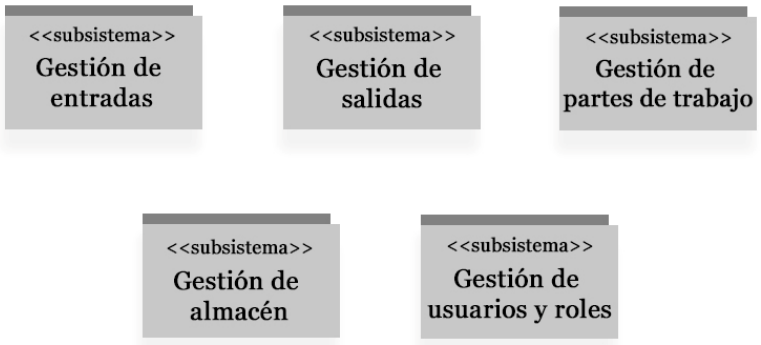



*Figura 10. Subsistemas de la aplicación web*

Como podemos ver en la Figura 16, en nuestro proyecto se han identificado cinco subsistemas distintos con tareas y funcionalidades únicas para cada uno de ellos. Una vez hemos sido capaces de identificar que subsistemas puede tener nuestro proyecto es momento de pasar al siguiente punto.

Antes de empezar con los detalles cabe decir que, en este proyecto se van a utilizar técnicas que han sido ampliamente probadas como eficaces en el sector, no obstante, es importante tener en cuenta que las que se muestran en este trabajo de fin de grado no son las únicas que existen

#### *4.2.1.1. Definición de actores*

En la tarea de definición de actores nuestro trabajo consistirá en definir de la manera más exacta posible los tipos de personas que van a intervenir en nuestro sistema, así como detallar que función tienen en él y cómo tienen que interactuar con el mismo.

En la realización de este proyecto se han identificado con la colaboración del cliente a tres tipos de actores, operarios, encargados y por último Administradores. Estos actores pueden realizar distintas tareas en el sistema y tener distintos privilegios, posteriormente se adjuntan los documentos que define con más exactitud a los actores.

| **ACT-01** | **Operario** |
|---|---|
| **Descripción** | Este actor representa a los trabajadores del almacén que gestionan las entradas y salidas de este. |
| **Comentarios** | ninguno |

| **ACT-02** | **Encargado** |
|---|---|
| **Descripción** | Este actor representa a los encargados del almacén, estos pueden realizar todas las tareas que realizan los operarios y también pueden insertar algunos parámetros de la gestión del almacén. |
| **Comentarios** | ninguno |

| **ACT-03** | **Administrador** |
|---|---|
| **Descripción** | Este actor representa a los administradores directos, son el cargo con más permisos del sistema, estos actores pueden realizar todas las acciones que realizan los operarios y los encargados. Adicionalmente pueden modificar la estructura de seguridad del almacén. |
| **Comentarios** | ninguno |

Como podemos ver, estos documentos cuentan con un identificador numérico de actor, con un nombre distintivo y con una breve introducción que detalla con más precisión su cometido en la aplicación.

#### *4.2.1.2. Casos de uso del sistema*

Una vez conocemos quién o quiénes van a ser los individuos que van a interactuar en nuestra aplicación, es el momento de identificar qué tareas se pueden realizar en nuestro sistema. Es en este apartado cuando los casos de usos entran en escena, y para entenderlos mejor vamos a ver un poquito de historia al respecto.

Los casos de uso son una técnica para la especificación de requisitos funcionales propuesta inicialmente en (Jacobson, 1999) y que actualmente forma parte de la propuesta UML (Booch, 1999).

Un caso de uso es la descripción de una secuencia de interacciones entre el sistema y uno o más actores en la que se considera al sistema como una caja negra y en la que los actores obtienen resultados observables.

Para este caso específico se propone rellenar plantillas en las que las interacciones se numeran siguiendo las propuestas de (Cockburn, 1997), (Schneider y Winters, 1998) y (Coleman 1998). (Durán Toro and Bernárdez Jiménez, 2002)

Para este proyecto en específico se han detectado más concretamente los casos de uso siguientes:

- UC-01 Insertar entradas
- UC-02 Insertar y procesar salidas
- UC-03 Consultar estado del almacén
- UC-04 Reubicar producto
- UC-05 Insertar productos
- UC-06 Consultar productos
- UC-07 Consultar los productos que se han ubicado
- UC-08 Consultar ubicación
- UC-09 Insertar tipos de ubicaciones
- UC-10 Consultar tipos de ubicaciones
- UC-11 Consultar los logs
- UC-12 Consultar avisos del almacén
- UC-13 Ubicar producto

- UC-14 Consultar entradas
- UC-15 Consultar salidas
- UC-16 Procesar entradas
- UC-17 Anular entradas
- UC-18 Dar de alta usuario
- UC-19 Dar de baja usuario
- UC-20 Asignar rol usuario
- UC-21 Quitar rol usuario
- UC-22 Asignar permisos de página a rol
- UC-23 Quitar permisos de página a rol
- UC-24 Eliminar entrada
- UC-25 Eliminar tipos de ubicaciones
- UC-26 Insertar ubicaciones
- UC-27 Eliminar ubicaciones
- UC-28 Insertar parte de trabajo
- UC-29 Modificar parte de trabajo
- UC-30 Eliminar parte de trabajo

Atendiendo a que resultaría demasiado extenso insertar todas las plantillas realizadas en este trabajo de fin de grado se ha decidido como se ha realizado en otros apartados insertar tan sólo una de las treinta plantillas realizadas a modo de ejemplo.

| **UC-16** | **Procesar entradas** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas | |
| **Requisitos asociados** | • IRQ-01 Información sobre entradas | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente procesar una entrada del almacén* | |
| **Precondición** | Para procesar una entrada esta debe existir con anterioridad | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de entradas |
| | 2 | El sistema carga la tabla con las entradas existentes en el sistema |
| | 3 | El empleado pulsa el botón de procesar entrada |
| | 4 | El sistema despliega una ventana en la que el empleado tiene que introducir la cantidad de productos recibidos |
| | 5 | El empleado confirma el procesamiento de la entrada |
| | 6 | El sistema inserta la información en el sistema y realiza los casos de uso UC-05 y UC-13 |
| **Postcondición** | Se ha procesado una entrada | |
| **Excepciones** | **Paso** | **Acción** |
| | 5 | Si el empleado introduce algún dato incorrecto o inconsistente el sistema se lo notifica |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 6 | 3 segundos |
| **Frecuencia** | Aprox. 2 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

### *4.2.1.3. Diagramas de casos de uso*

El siguiente paso será plasmar los casos de uso anteriores en un formato mucho más gráfico, y para ello vamos a utilizar una herramienta conocida como diagramas de casos de uso. La función de estos diagramas es de simplificar los casos de uso y hacerlos mucho más visuales.

Los diagramas de casos de uso constan de distintos elementos para su representación. Los actores que intervienen en los casos de uso aparecen como pequeñas figuras con forma de persona que se relacionan mediante flechas con los casos de uso con los que se supone que son capaces de interactuar.

Las flechas de las relaciones entre actores y casos de uso indican con sus puntas en qué dirección fluye la información. También existen como veremos flechas especiales que incluyen o desencadenan otros casos de uso.

Para este proyecto se han realizado cuatro diagramas de uso, uno por cada subsistema que se ha detectado. Como en secciones anteriores, debido al volumen de casos de uso del proyecto sólo se va a mostrar un diagrama a modo de ejemplo. Para este caso concreto se ha decidido mostrar el diagrama del subsistema de gestión de entradas, ya que tiene algunas de las técnicas que ya se han comentado.

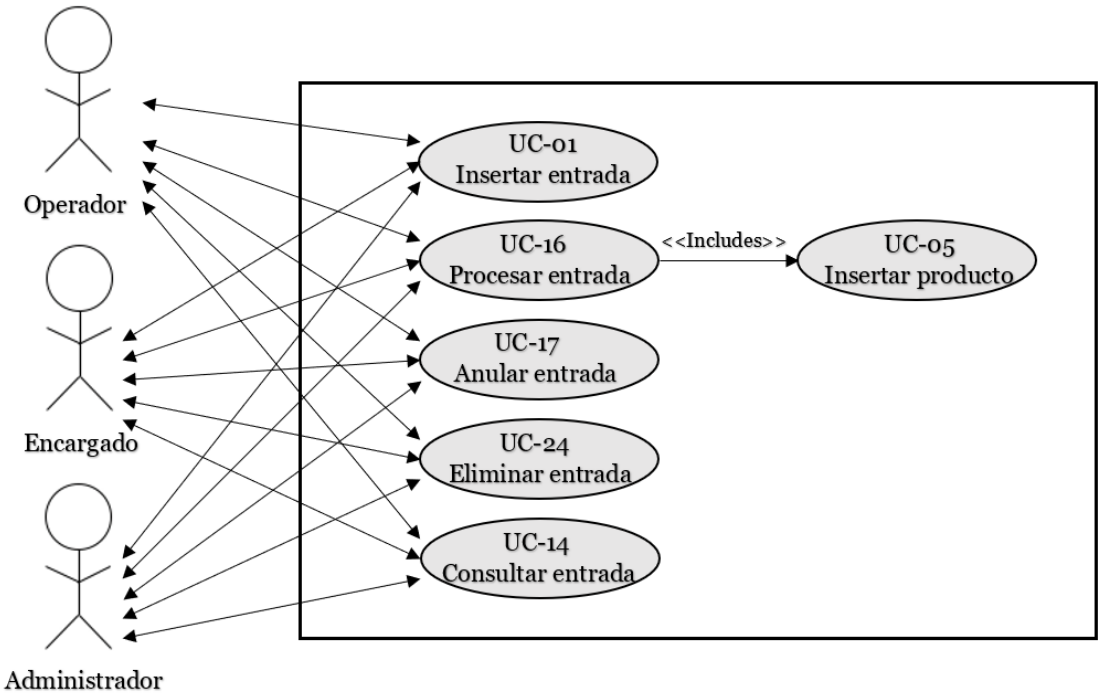



*Figura 11. Diagrama de uso correspondiente al subsistema de gestión de entradas*

Como podemos ver en la Figura 17, en este subsistema interactúan operarios, encargados y administradores por igual. Si nos fijamos más en detalle, podemos apreciar que el caso de uso UC-05 se relaciona utilizando la flecha <<includes>>. La razón es que este caso de uso es realizado por el sistema de manera autónoma una vez se haya realizado el caso UC-16.

### 4.2.2. Requisitos no funcionales

Los requisitos funcionales son todos aquellos que hacen referencia a elementos que no tienen que ver con la información de la aplicación. No obstante, siguen siendo requisitos que forman parte de nuestra aplicación y que deben ser tenidos en cuenta en la especificación.

Algunos tipos de requisitos que pueden encajar en este apartado como requisitos no funcionales son:

- Requisitos de comunicaciones del sistema
- Requisitos de interfaz de usuario
- Requisitos de fiabilidad
- Requisitos de entorno de desarrollo
- Requisitos de portabilidad

Para este proyecto se han detectado tres requisitos no funcionales que deben ser tenidos en cuenta y consideradas restricciones.

| **NFR-01** | **Entorno de ejecución** |
|---|---|
| **Objetivos asociados** | - |
| **Requisitos asociados** | - |
| **Descripción** | El sistema debe ser ejecutado en un entorno físico que proporciona el cliente. |
| **Comentarios** | En este caso se revisa el servidor que ofrecen y es un servidor completamente capaz de ejecutar la aplicación sin ningún tipo de impedimento |

| **NFR-02** | **Tolerante a errores** |
|---|---|
| **Objetivos asociados** | - |
| **Requisitos asociados** | - |
| **Descripción** | El sistema debe ser capaz de ofrecer distintas soluciones en caso de que se llegara a producir un error en el sistema o cualquier otro tipo de incidente. Se valora la disponibilidad y la tolerancia a fallos. |
| **Comentarios** | - |

| **NFR-03** | **Compatibilidad** |
|---|---|
| **Objetivos asociados** | - |
| **Requisitos asociados** | - |
| **Descripción** | El sistema debe ser desarrollador con intención de ser ejecutado en una máquina con licencia de Windows Server ya que es el tipo de máquinas con las que cuenta nuestro cliente. |
| **Comentarios** | Debido a que la aplicación debe ser ejecutada obligatoriamente en Windows, la ventaja de multiplataforma deja de ser un elemento de peso en la decisión de las herramientas. |

### 4.2.3. Requisitos de información

En este punto vamos a identificar los requisitos de información de nuestra aplicación, es decir, identificar toda la información que deberá almacenar nuestro sistema de información teniendo en cuenta los objetivos definidos con anterioridad.

En resumen, en este apartado debemos identificar la información relevante para el cliente, si existen conflictos en el almacenaje de esta información, incluyendo también de la mano del cliente las restricciones o reglas de negocio que afectan a los requisitos de información de nuestro sistema.

Para realizar este cometido vamos a rellenar una plantilla que parte de los conceptos más generales para posteriormente ir detallándolos hasta obtener todos los datos relevantes. En este caso se han detectado los siguientes requisitos de información en nuestro sistema:

- IRQ-01 Información sobre entradas
- IRQ-02 Información sobre salidas
- IRQ-03 Información sobre la trazabilidad de los productos
- IRQ-04 Información sobre stock
- IRQ-05 Información sobre partes de trabajo

Por razones de espacio solamente se va a presentar una de estas plantillas a modo de ejemplo, las plantillas restantes pueden ser encontradas en el anexo de este trabajo de fin de grado.

| **IRQ-01** | **Información sobre entradas** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas de los productos | |
| **Requisitos asociados** | • UC-01 Insertar entradas<br>• UC-17 Anular entradas<br>• UC-13 Ubicar producto<br>• UC-14 Consultar entradas<br>• UC-16 Procesar entradas | |
| **Descripción** | *El sistema debe almacenar la información relativa a las entradas que se producen en el almacén, en este caso debe almacenar estos datos:* | |
| **Datos específicos** | • En qué almacén se encuentra<br>• En qué ubicación se encuentra<br>• La cantidad de productos asociada a la entrada<br>• Si está procesada<br>• Quién la ha procesado<br>• Si está anulada<br>• Quién la ha anulado<br>• Cuándo fue introducida | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| | 1.5 años | 4 años |
| **Ocurrencias simult.** | **Medio** | **Máximo** |
| | 10 | 20 |

| | |
|---|---|
| **Estabilidad** | Alta |
| **Comentarios** | *Se han introducido los requisitos UC-13, UC-16, UC-17 debido a que el sistema una vez procesa las entradas ubica automáticamente los productos y este también puede ser anulado.* |

| **CRQ-01** | **Relación entre entradas y ubicaciones** |
|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas de los productos |
| **Requisitos asociados** | • IRQ-01 Información sobre entradas<br>• UC-16 Procesar entradas<br>• UC-13 Ubicar producto |
| **Descripción** | Los datos almacenados en el sistema referentes a las entradas deben cumplir siempre que: *cuando una entrada se encuentra procesada debe ubicarse inicialmente en la ubicación de entrada para posteriormente ser ubicada dónde recomiende el almacén.* |
| **Estabilidad** | Alta |
| **Comentarios** | ninguno |

# 5. Propuesta de desarrollo

## 5.1. Elección de herramientas para el proyecto

Para la elección de las herramientas que vamos a emplear en nuestro proyecto, se ha utilizado toda la investigación realizada en el capítulo dos de este trabajo de fin de grado. Con esta información, se va a explicar que herramienta o lenguaje de los expuestos anteriormente van a ser utilizados en la realización de la aplicación y por qué.

Como se ha realizado en secciones anteriores, se comenzará con la elección de las herramientas más cercanas al cliente y progresivamente se irá acercando más al lado del servidor o backend.

**Interfaz o frontend**

En este caso y al ser una aplicación web, se ha considerado que lo más adecuado es seleccionar las herramientas más estandarizadas en este sector. Cabe recalcar que, si se hubiese decidido desde el principio la realización de una aplicación sin soporte web, la elección hubiese sido mucho más compleja debido a la gran cantidad de módulos existentes para desarrollar interfaces gráficas.

Para la estructura de la interfaz se ha decidido utilizar HyperText Markup Language o HTML. Como hemos comentado previamente, sus grandes ventajes residen en que es una herramienta sencilla, intuitiva, tremendamente potente y con capacidad de realizar modificaciones instantáneas. Actualmente existen alternativas distintas a este lenguaje de marcado como BBC o XML, no obstante, el más extendido con diferencia en el sector profesional es HTML.

Los factores anteriores y la conveniencia profesional de este lenguaje hacen que lo más sensato sea seleccionar HTML como la herramienta que dará forma a la estructura de esta aplicación web.

Para la presentación de nuestra aplicación web se ha decidido utilizar Cascade Style Sheet o CSS porque es una herramienta que se complementa muy bien con HTML y nos permite reducir la cantidad de código con sus clases. Adicionalmente, se pueden realizar grandes cambios con pocas líneas de código y combinada con un framework como Bootstrap nos va a permitir tener páginas web muy completas visualmente.

Por último, para el dinamismo y la lógica visual de la aplicación se ha decidido utilizar Javascript como lenguaje principal por su rapidez y el gran ecosistema de frameworks que tiene a su alrededor. Como framework de apoyo a Javascript se ha decidido utilizado AngularJS porque es un framework desarrollado por Google, ampliamente utilizado y que permite la creación de componentes que pueden ser reutilizados en el código.

### Interfaz de programación de aplicaciones o API

Antes de profundizar en el lenguaje de programación de nuestra API vamos a hablar del tipo de API que va a utilizar nuestra aplicación. Teniendo en cuenta que finalmente se va a utilizar el protocolo HTTP para la comunicación y se busca la mayor sencillez posible, se ha decidido que nuestra API será de tipo Rest.

Por otro lado, teniendo en cuenta que el cliente requiere que esta aplicación corra exclusivamente sobre Windows, parece una decisión coherente seleccionar ASP.NET Core como framework para nuestra API ya que este está desarrollado directamente por Microsoft y goza de facilidades con su propio sistema operativo. Finalmente, se ha decidido utilizar C# como lenguaje que acompañe a ASP.NET Core por su sencillez y teniendo en cuenta que ambos utilizan programación orientada a objetos.

### Base de datos

Teniendo en cuenta que nuestro cliente quiere que trabajemos con Windows y con una base de datos relacional, nuestras opciones se reducen ya que se recomienda Oracle, SQL Server o MySQL como las bases de datos más compatibles y sencillas de utilizar con este sistema operativo.

En este caso, nuestro cliente posee licencias de SQL Server, es por este motivo por el que vamos a utilizar SQL Server como nuestra base de datos. No obstante, es importante recalcar que, si nuestro cliente no tuviera licencias ya adquiridas, la elección lógica sería MySQL ya que es una base de datos muy potente y totalmente gratuita.

### Servidores web HTTP

Como ha ocurrido en puntos anteriores, si se tiene en cuenta que nuestra aplicación web tiene que ser ejecutada en Windows y adicionalmente sabemos que nuestra comunicación va a ser realizada mediante el protocolo HTML, es razonable elegir para este cometido el propio servidor web que desarrolla Microsoft.

El servidor web de Microsoft es Microsoft Internet Information Services (IIS), se ha seleccionado este servidor por su facilidad de manejo y la gran seguridad que ofrece respecto a otros servidores, con IIS podemos tener en cuestión de segundos una página web plenamente operativa y es una herramienta que viene por defecto con Windows Server.

### Servidor

Previamente hemos mostrado distintos tipos de servidores entre los que se encontraban servidores no físicos y servidores físicos en los que podíamos elegir según la escalabilidad que tuviésemos en mente, cabe decir que cualquiera de las opciones no físicas o físicas expuestas con anterioridad es suficientemente potente como para ejecutar el proyecto web planteado en este trabajo de fin de grado.

A pesar de todo esto, nuestro cliente posee ya distintos servidores físicos que utilizan para cuestiones de desarrollo y producción. Es por este motivo que vamos a utilizar el servidor que ellos nos habilitan para ejecutar y desarrollar nuestra aplicación, en este caso este es el servidor que nos proporcionan.

*Figura 12. Servidor tipo rack DELL PowerEdge R420* (DELL, 2020)

| **DELL PowerEdge R420** | |
|---|---|
| **Procesador** | Intel Xeon E5-2407 v2 - 2.40Ghz (8 CPUs) |
| **Memoria principal** | 65 GB DDR3 |
| **Placa base** | R420 Motherboard |
| **Tarjeta de red** | Broadcom® 5720 Dual Port 1Gb LOM |
| **Memoria secundaria** | 1 TB |
| **Alimentación** | 550W Cableada |
| **Descripción** | Ninguno |

Este es un servidor de gama media-alta con unas especificaciones realmente buenas y muy por encima de las necesidades de nuestra aplicación. Es importante recalcar que, si hubiésemos tenido la opción de elegir, basándonos en la investigación realizada anteriormente en este trabajo de fin de grado, nos hubiésemos decantado por un servidor de gama baja tipo torre o tipo rack con un rango de precio entre los 500 – 700 euros, también se podría optar por cualquier plan más económico de iniciación que ofrecen empresas dedicadas a servicios en la nube o servidores virtuales privados.

Adquirir este servidor para este proyecto exclusivamente sería totalmente desproporcionado y triplicaría fácilmente el presupuesto entre el que deberíamos movernos para un proyecto de estas características.

**Programario a utilizar**

Aparte de todas las herramientas presentadas previamente, se necesitan herramientas de apoyo como editores de código, herramientas de gestión para la base de datos entre otras. Estas son las herramientas adicionales que se han utilizado para la creación de este proyecto.

- **Visual Studio Community 2019**: Utilizado como editor de código para la interfaz y la API.
- **SQL Server Management Studio 2019:** Gestión de bases de datos SQL Server.
- **Advanced Rest Client (ARC):** Programa para realizar peticiones HTTP a la aplicación API.
- **Visual Studio Code:** Utilizado como editor de código alternativo a Visual Studio.
- **Conexión a Escritorio Remoto de Windows:** Programa incorporado de Windows utilizado para la conexión con el servidor.

## 5.2. Algoritmo de almacenaje

Teniendo en cuenta todas las ventajas y desventajas expuestas en el estado del arte sobre las diferentes estrategias de almacenamiento que existen, se ha decidido utilizar una estrategia de posiciones aleatorias escogidas por el empleado debido a que se está informatizando un almacén y aunque sea un almacén pequeño se puede exprimir mucho más las ubicaciones y el espacio que tiene.

Este tipo de estrategias va a dar total libertad a los operarios del almacén para ubicar productos y no va a tener ningún tipo de restricción de tipo de producto o similar, por tanto, el sistema no va a reservar posiciones para productos específicos pudiendo así utilizar la máxima capacidad de almacenaje que este tiene.

## 5.3. Flujos de trabajo de los procesos más críticos

Uno de los métodos más interesantes para visualizar y especificar diferentes procesos de negocio son los diagramas de flujo, este tipo de diagramas se han utilizado muchas veces para definir procesos durante la realización del grado cursado. En concreto, la notación más interesante y que más se ha utilizado es Business Process Model and Notation (BPMN).

La notación BPMN proporciona una notación gráfica para especificar modelos de negocio en un diagrama de procesos de negocio (BPD). Su objetivo principal es el de dar soporte al modelado de procesos de negocio proporcionando una notación estándar que sea comprensible para todos los usuarios, pero que también sea capaz de representar procesos complejos para usuarios técnicos (OMG, 2006).

El objetivo principal de BPMN es proporcionar una notación que sea fácil de entender por todos los usuarios, desde los analistas de negocio que crean los bocetos iniciales de los procesos, hasta los desarrolladores responsables de implantar la tecnología que hará funcionar estos procesos, finalizando con los usuarios del negocio que van a gestionar y monitorizar esos procesos (OMG, 2006).

Dado que ya conocemos a grandes rasgos que es la notación BPMN y para qué sirve, es momento de identificar los procesos críticos en la aplicación. Se han identificado estos procesos como procesos críticos.

- Procesar entradas
- Reubicar producto
- Insertar y procesar salida

Por cuestiones de densidad, se va a insertar sólo el diagrama de procesar entradas ya que es considerado el proceso más crítico en el sistema, el resto de los diagramas pueden ser encontrados en el anexo.

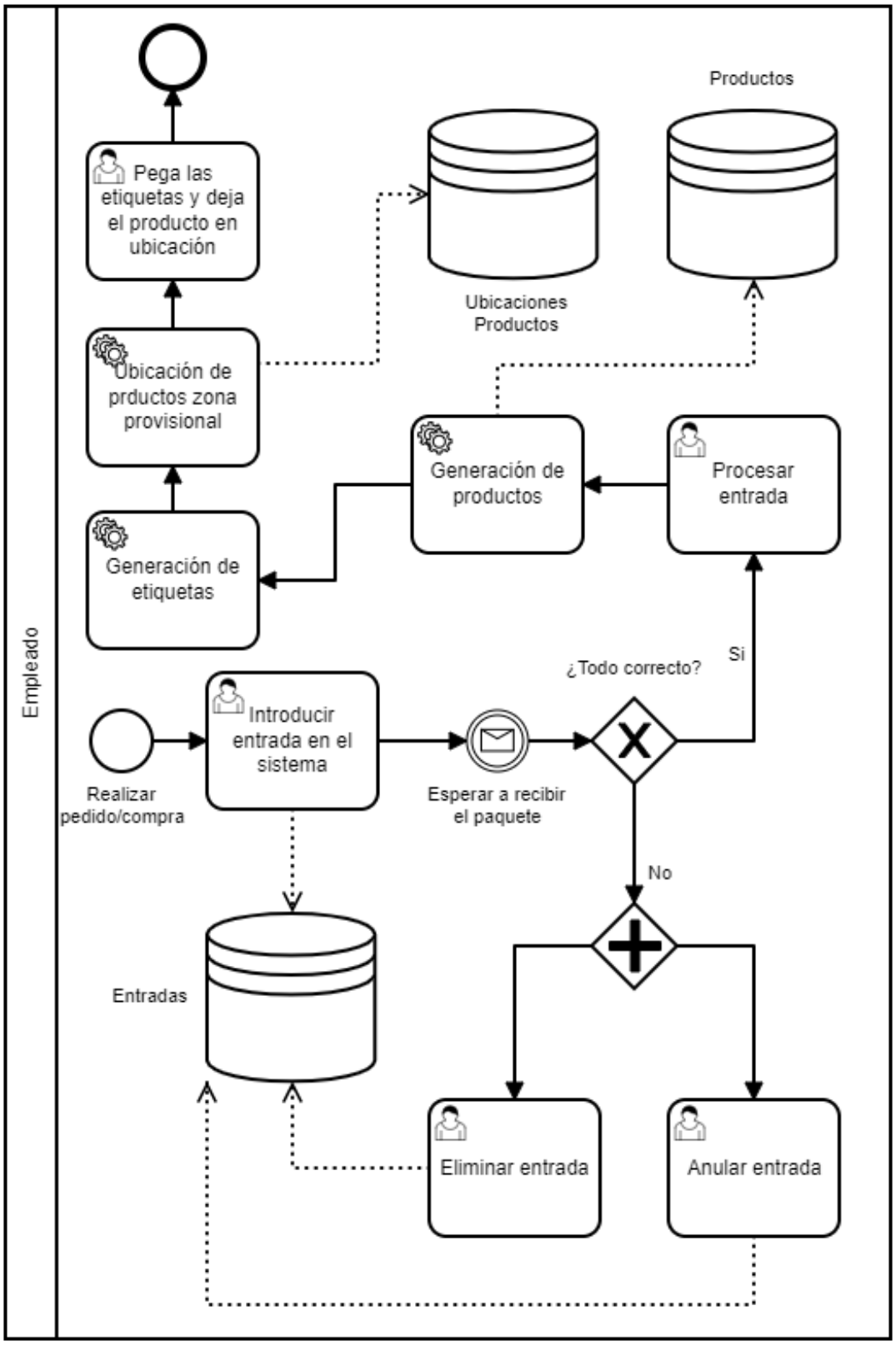



*Figura 13. Diagrama de flujo de ProcesarEntrada*

## 5.4. Bocetado de la interfaz

Para la realización de los bocetos de una aplicación debemos tener en cuenta que deben ser sencillos y no reflejar el estado final de la aplicación ya que podría construir en el cliente una idealización de cómo debería ser la aplicación en su estado final, esto es un error ya que durante el desarrollo de una aplicación se producen muchos cambios y ajustes.

Las siguientes figuras muestran los bocetados de la página para iniciar sesión y otro que refleja una posible composición de una página estándar de la aplicación para visualizar los datos en una tabla. Estos bocetos vienen acompañados con algunas indicaciones sobre el tipo de estructura HTML que se podrían utilizar y algunas notas que indican para que pudiera ser utilizada cada sección.

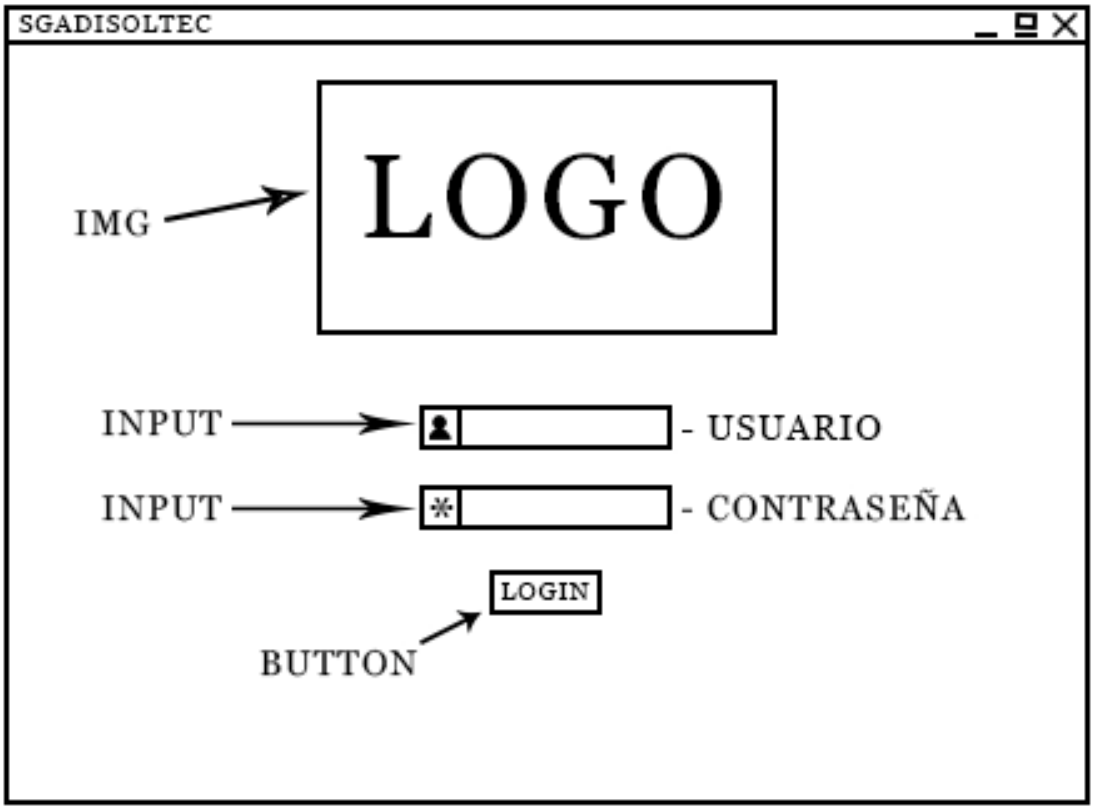



*Figura 14. Ejemplo de interfaz de login para SGADISOLTEC*

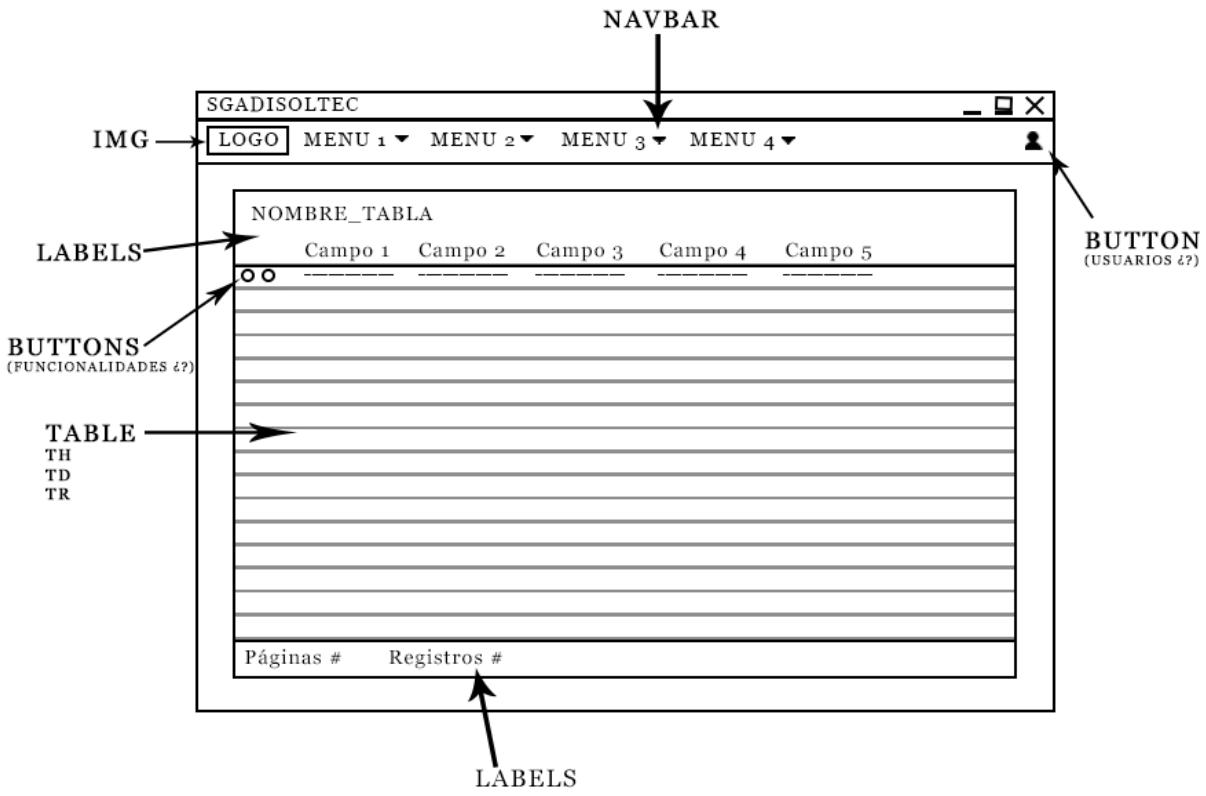



*Figura 15. Ejemplo de página para SGADISOLTEC*

## 5.5. Diseño de la base de datos

En la actualidad, la mayoría de los sistemas almacenan mucha información y requieren un sistema dedicado al almacenamiento y correcto tratamiento de esta información. Este caso no es distinto al resto, y, al tratarse de un almacén, necesitamos una estructura de datos que sea adecuada y capaz de almacenar todos los datos que nos permitirán virtualizar el almacén para tomar decisiones sobre sus datos.

Existen muchas herramientas para definir herramientas. No obstante, para este trabajo de fin de grado creemos conveniente aplicar técnicas que ya hemos desarrollado con anterioridad en el grado y con las que tenemos cierta familiaridad sabiendo que son eficaces.

### 5.5.1. Descripción del sistema de información

Lo primero que vamos a hacer será describir en un breve texto cuáles son los elementos de información que debe tener nuestra aplicación. Esta tarea puede parecer que carece de utilidad, pero creemos que es más bien todo lo contrario. Detallar el sistema de información en palabras nos ayudará a trazar una guía sobre la que podremos empezar a construir nuestro sistema de información y nos ayudará a pasar menos cosas por alto.

En nuestro proyecto se reciben muchos bultos debido a que es una empresa que ofrece a otras empresas soluciones tecnológicas de todo tipo, desde cambiar una pantalla hasta ofrecer software de gestión avanzada de procesos. Es por este motivo por el que se plantean virtualizar su almacén y registrar todos los productos que tienen para llevar un control más preciso.

El almacén debe ser capaz de registrar entradas de bultos en el sistema. De cada modelo se guardan datos como la cantidad de productos, la fecha de recepción, quién las ha introducido y también debe ser capaz de poder eliminarlas, anularlas o procesarlas.

Cuando una entrada se procesa, se generan automáticamente los productos acordes a la cantidad que tenía registrada la entrada. Una vez creados los productos, estos se ubican automáticamente en una zona predeterminada para que posteriormente sean reubicados al lugar recomendado por el almacén o el que desee el empleado.

Por otro lado, los empleados sacan del almacén de manera habitualmente productos para ser instalados, es por este motivo por el que el sistema debe ser capaz de generar salidas. Estas salidas son la manera de notificarle al almacén que sale un producto de sus ubicaciones. Para este cometido, el sistema debe almacenar dónde se encontraba el producto, la cantidad de productos que salen e información sobre el estado de la propia salida.

Las ubicaciones del almacén deben ser escalables en todo momento y el almacén debe ser capaz de soportar ampliaciones sin ningún problema. De la misma manera, el sistema debe ser capaz de entender que existen ubicaciones que por sus prestaciones se reservarán a determinados productos.

Por otra parte, el sistema debe ser capaz de almacenar e imprimir todos los partes de trabajo que los empleados realizan a modo de factura para sus clientes. Estos partes de trabajo deben contener tanto información relativa al trabajador como información relativa al trabajo realizado, horario en el que se ha trabajado, total de horas, descripción del trabajo realizado, nombre del trabajador y su firma.

### 5.5.2. Esquema relacional

El esquema relacional permite plasmar de una manera visual toda la lógica de la base de datos, de esta manera, podemos ver las tablas que conforman el sistema, las relaciones que tienen entre sí, los datos que se almacenan e incluso los tipos de datos definidos para cada campo. Para entender correctamente este esquema relacional hay que tener en cuenta que se ha utilizado NN para los campos no nulos y N para los campos nulos, el resto del esquema relacional sigue la estructura de un esquema relacional convencional. Estos son los esquemas relacionales de la aplicación para la gestión de almacenes y posteriormente el esquema relacional de los partes de trabajo.

Se ha incluido una tabla llamada Reubicaciones que no aparece en el esquema y se utiliza únicamente para registrar en qué momento se cambian los productos de ubicación. Por otro lado, por motivos de extensión se anexa al final del proyecto el esquema relacional para el sistema de usuarios.

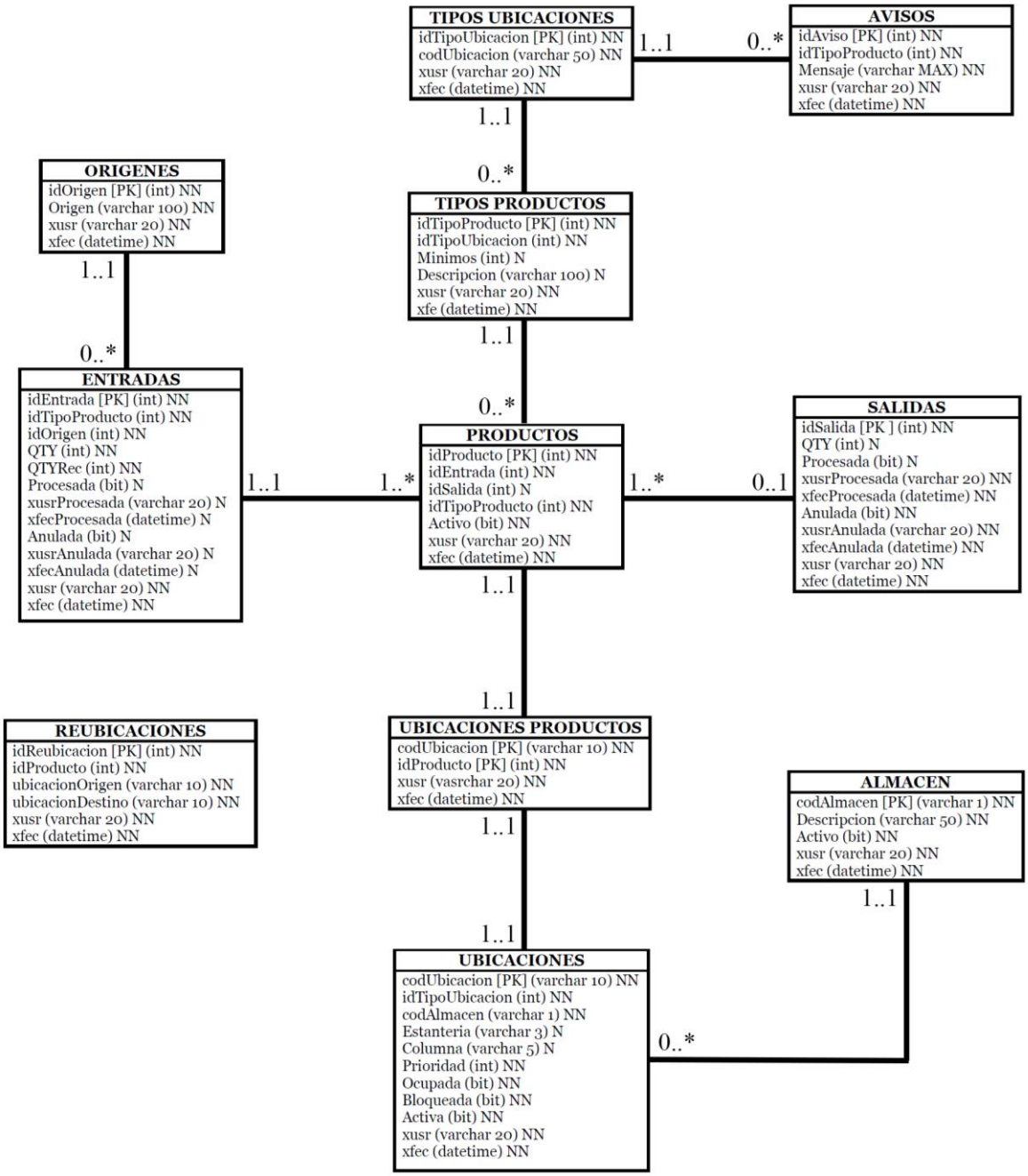



*Figura 16. Esquema relacional de SGADISOLTEC*

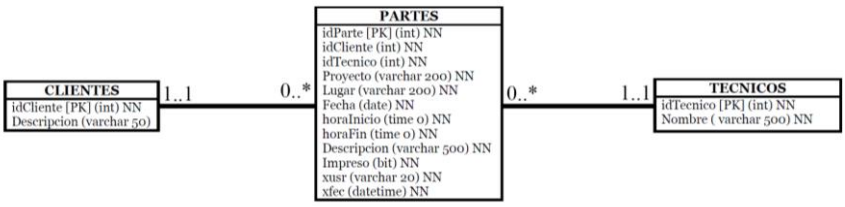



*Figura 17. Esquema relacional de partes de trabajo*

# 6. Desarrollo de la solución

## 6.1. Desarrollo de la aplicación web

Una vez tenemos un servidor y todas las especificaciones necesarias realizadas y contrastadas con el cliente, es momento de empezar con el desarrollo de la aplicación web.

Es importante seguir cierto orden para desarrollar una aplicación. Sobre todo, se recomienda no abarcar demasiados frentes a la vez; una estrategia más sensata sería dividir los problemas y afrontarlos uno a uno de manera secuencial.

Para nuestra aplicación, primero prepararemos nuestra estructura de trabajo, posteriormente elaboraremos el proyecto, haremos nuestra base de datos con la estructura propuesta anteriormente, crearemos y pondremos en funcionamiento nuestra API y por último nos pondremos con la parte visual y dinámica de la aplicación web.

### 6.1.1. Preparación del entorno de trabajo

En este apartado es en el que deberíamos realizar la instalación del sistema operativo que hayamos seleccionado previamente. En nuestro caso y por requerimiento de la empresa, la aplicación tiene que funcionar con Windows.

Si sumamos esto a que la propia empresa nos presta un servidor suyo, nos encontramos en la situación de que el sistema ya está instalado y configurado para que nuestra aplicación pueda funcionar correctamente.

Otro punto importante en nuestro entorno de trabajo es la estructura y ordenación de los ficheros de nuestra aplicación, pues existen muchas maneras de estructurar los proyectos y todas son válidas. Para este proyecto se ha decidido trabajar con la siguiente estructura de documentos por comodidad.

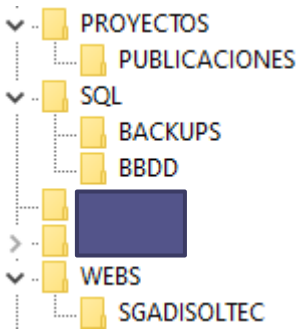



*Figura 18. Estructura de documentos de la aplicación*

Como se puede ver en la figura esta es la estructura de datos que se ha creado para la realización de este proyecto, vamos a hacer una breve descripción de cada una:

- Proyectos: En esta carpeta es dónde se va a almacenar la solución del proyecto que crearemos utilizando Visual Studio. Se ha creado también una subcarpeta para hacer publicaciones de la API.
- SQL: En esta carpeta se almacenarán los archivos de la base de datos y también se ha creado una carpeta para almacenar versiones anteriores de la base de datos.
- Webs: Por último, en esta carpeta es dónde se almacenará la aplicación web. Como puede darse el caso de que tengamos distintas páginas web funcionando en la misma máquina, se ha creado una carpeta exclusiva para los documentos web del proyecto.

### 6.1.2. Creación de la solución

El principal objetivo de crear un proyecto o una solución es tener todos los documentos en un mismo sitio, de esta manera, podemos modificar cualquier documento de nuestra aplicación, desde la interfaz hasta la API sin cambiar de programas u ordenadores.

Para la creación del proyecto o solución se ha utilizado en este caso Visual Studio Community, la versión 2019 de este conocido editor de código. En las figuras que se muestran a continuación se explica brevemente como crear una solución para nuestra aplicación.

Lo primero una vez está abierto Visual Studio es ir al apartado Archivo y crear un nuevo proyecto como se muestra en la siguiente figura.

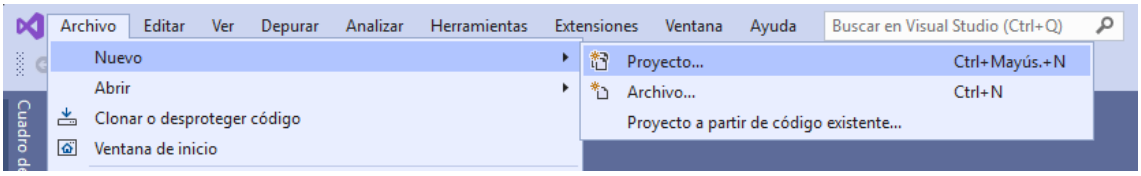



*Figura 19. Creación de un proyecto en Visual Studio*

Una vez hecho esto se desplegará otra ventana en la que Visual Studio nos va a permitir crear proyectos utilizando plantillas predefinidas, en este caso no necesitamos ninguna plantilla ya que nosotros estamos haciendo nuestra propia estructura de datos. Teniendo en cuenta esto, la opción que debemos elegir en el tipo de proyecto es solución en blanco.

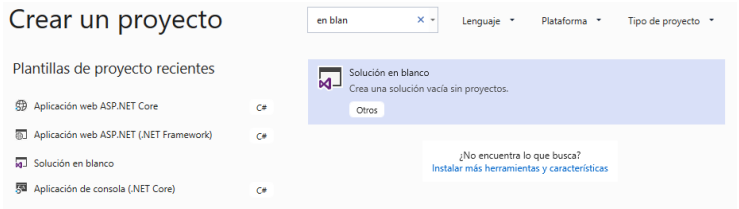



*Figura 20. Seleccionar el tipo de proyecto que se crea*

Posteriormente, Visual Studio nos pedirá el nombre de la solución que queremos crear y la ubicación de este, el nombre suele ser el mismo que el del proyecto y para la ubicación de la solución se va a utilizar la carpeta PROYECTOS creada en la estructura de datos en la sección anterior.

El siguiente paso es llenar la solución con nuestra página web, para ello damos botón derecho en nuestra solución, agregar y finalmente sitio web existente como se aprecia en la figura 21. En este punto VisualStudio nos consulta dónde está la página web que queremos agregar, como anteriormente se ha creado una carpeta para almacenar nuestra página web en WEBS utilizaremos esta ruta.

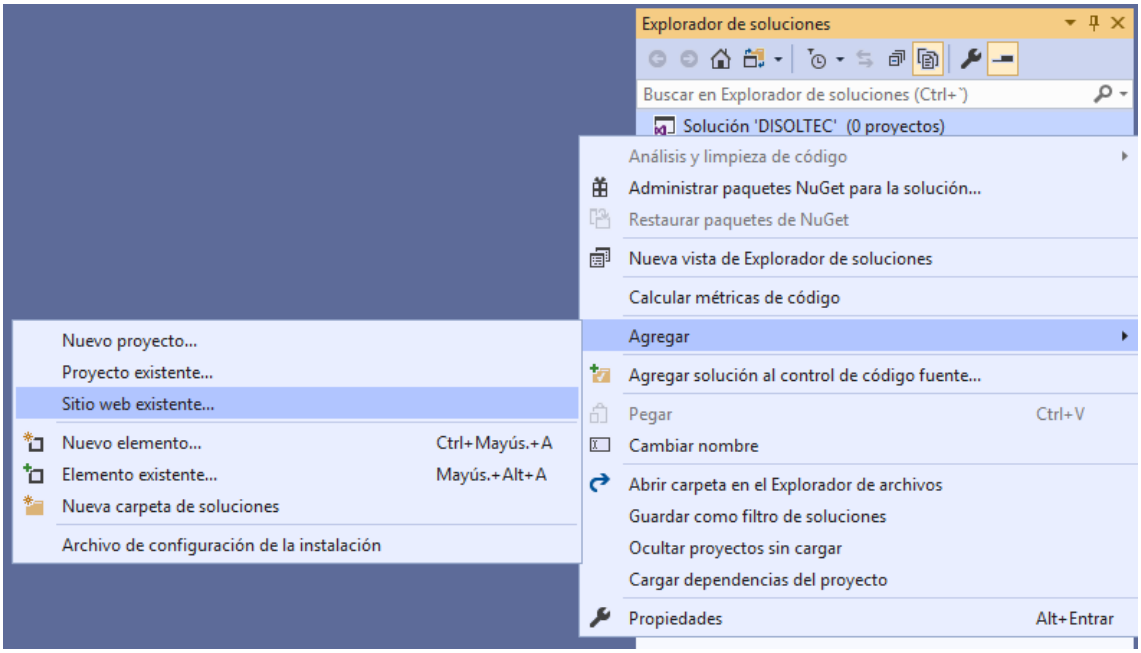



*Figura 21. Agregar sitio web existente en Visual Studio*

Después de esto ya se puede visualizar en nuestra solución nuestro sitio web y se pueden ver, modificar, crear o eliminar cualquier archivo que se encuentre dentro de nuestra carpeta de documentos web.

### 6.1.3. Creación de la RestAPI y configurar IIS

Para crear la interfaz de programación de aplicaciones o API vamos a utilizar del mismo modo que antes, una plantilla ya creada en Visual Studio para proyectos de ASP.NET Core diseñada para un servicio RESTful y que utiliza el protocolo HTTP para la comunicación.

Partiendo del punto anterior volvemos a crear un proyecto tal y como se ha creado anteriormente, a diferencia del apartado anterior, en este elegiremos tipo de proyecto Aplicación web ASP.NET Core como podemos ver en la figura 22.

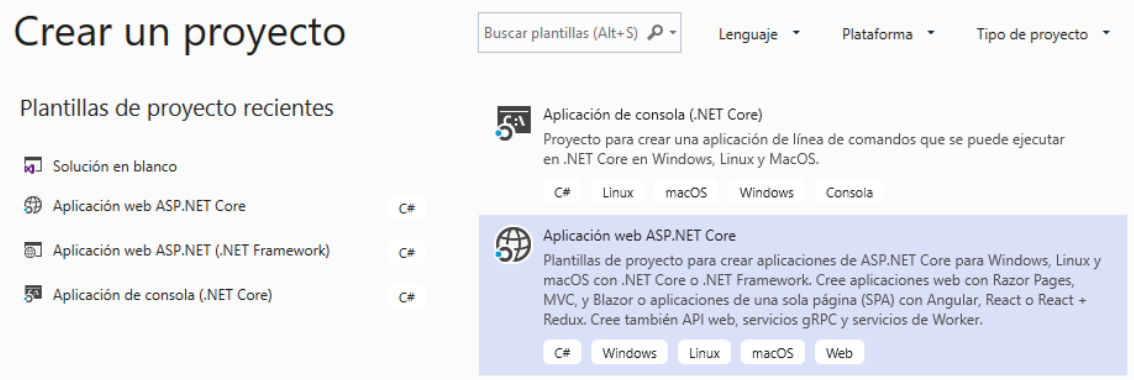



*Figura 22. Creación de Aplicación web ASP.NET Core*

Después de este paso, se nos despliega una ventana que nos dice dónde queremos que se cree el proyecto, como anteriormente se ha creado ya un proyecto se selecciona la opción agregar a solución para tener todo unificado en la misma solución.

En la siguiente ventana nos va a pedir que tipo de plantilla vamos a querer utilizar y que versión de ASP.NET Core, para este proyecto se ha dejado tal y como se muestra en la siguiente figura.

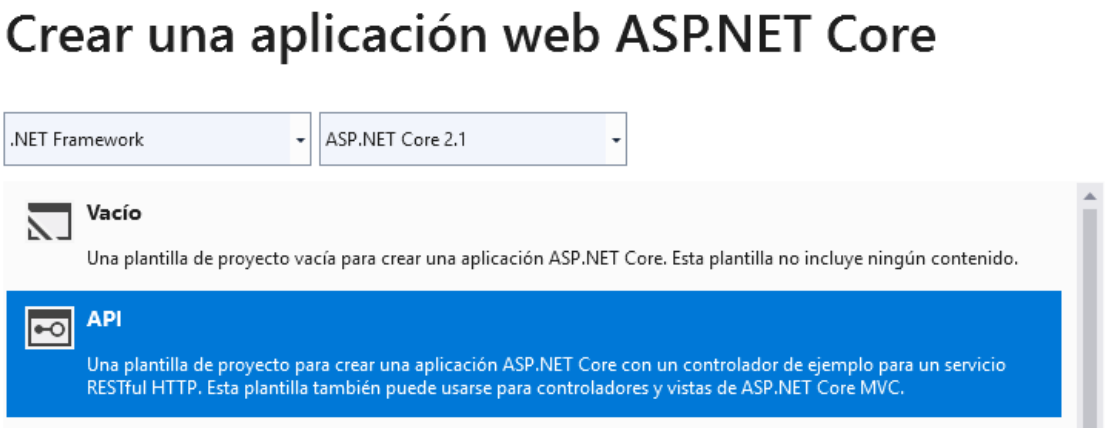



*Figura 23. Seleccionar plantilla para aplicación web ASP.NET Core*

Es importante recalcar que la elección de versión de ASP.NET Core debe coincidir con la versión que tengamos instalada en nuestro equipo o servidor, de lo contrario, nuestra API no va a ser capaz de funcionar correctamente y no podrá atender a las peticiones HTTP que reciba.

Lo único que resta antes de configurar IIS es publicar la API y copiarla en la carpeta que hemos creado previamente para ello, WEBS/SITIOWEB/RestAPI esto es debido a que la API va a estar contenida dentro de la propia estructura de datos dónde se almacena nuestro sitio web por comodidad y sencillez.

Para configurar nuestro sitio web en IIS lo primero que se debe hacer es crear un sitio web, para ello abrimos la aplicación, pulsamos botón derecho y agregar sitio web. Justo después de pulsar el botón se despliega una ventana en la que se tiene que rellenar la configuración de nuestro sitio web, en esta ventana tenemos que poner el nombre de nuestro sitio web y asignarle un puerto que esté disponible en nuestra red ya que este sitio web será ejecutada sólo dentro de nuestra red tal y como se aprecia en la figura 24.

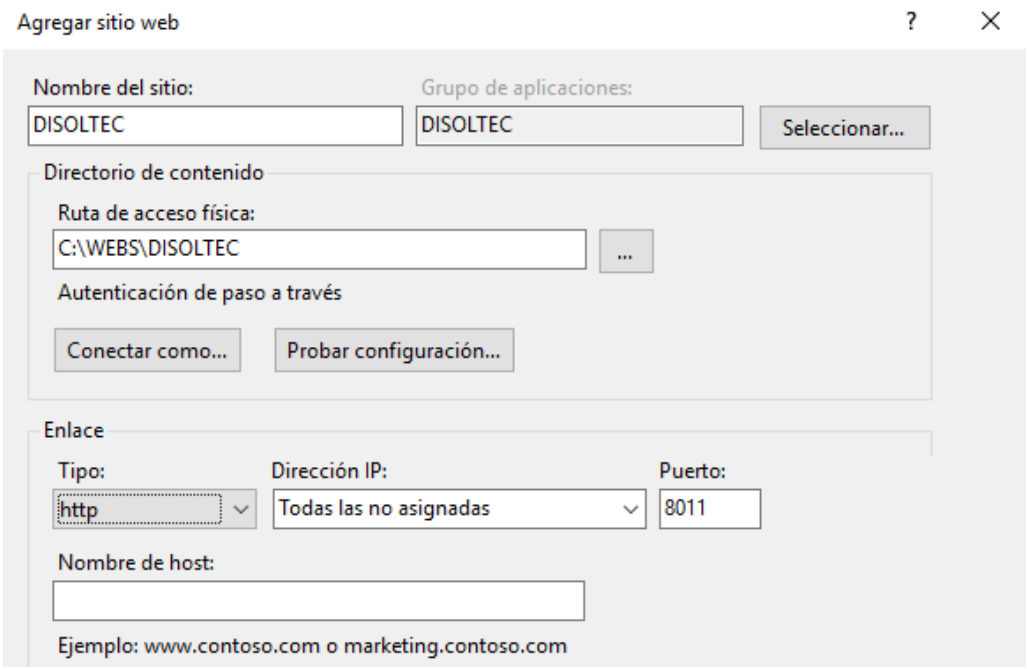



*Figura 24. Agregar sitio web en Internet Information Services*

Posteriormente, se crea un grupo de aplicaciones para que IIS detecte la carpeta RestAPI como una aplicación, para ello se pulsa botón derecho en grupo de aplicaciones y pulsamos agregar grupo de aplicaciones. Una vez se haya abierto la ventana de creación, rellenamos todos los parámetros como se indica en la siguiente figura.

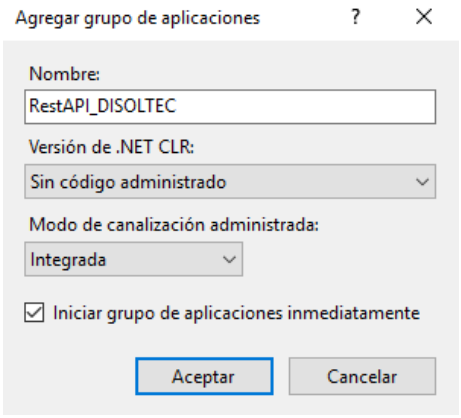



*Figura 25. Creación de un grupo de aplicaciones en Internet Information Services*

Finalmente, se asigna el grupo de aplicaciones creado con anterioridad a la carpeta que contiene nuestra RestAPI. Para realizar esta tarea, desplegamos nuestro sitio web y navegamos hasta la carpeta RestAPI, una vez localizada pulsamos botón derecho sobre ella seguido de convertir en aplicación, para finalizar se rellena la ventana emergente como se muestra a continuación.

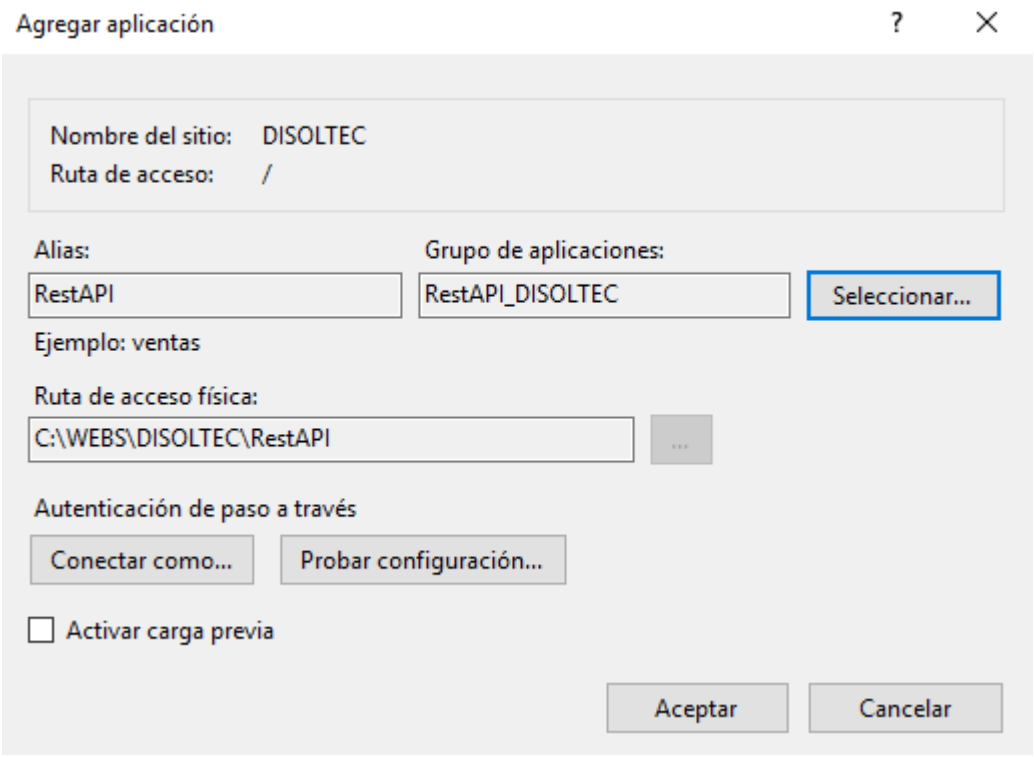



*Figura 26. Agregar carpeta como grupo de aplicaciones en Internet Information Services*

Llegados a este punto, se debería poder acceder a la aplicación web mediante el puerto que hayamos asignado y probar a hacer alguna petición HTTP a la API para comprobar que todo está correctamente configurado.

### 6.1.4. Creación de la BBDD

Para la creación de la base de datos se ha utilizado Microsoft SQL Server Management Studio en su versión de 2019, este programa se utiliza ampliamente para la gestión de bases de datos basadas en SQL Server. Para crear nuestra base de datos tenemos que iniciar sesión con las credenciales definidas en el momento de la instalación de SQL Server en el servidor.

Una vez nos hayamos identificado como administrador del sistema se pulsa botón derecho en la sección Databases seguido de New Database, esto nos despliega una ventana en la que podemos especificar parámetros como dónde se van a guardar los archivos de la base de datos y sus logs, el nombre de la base de datos, compatibilidades y muchas más opciones. Para este proyecto simplemente se han cambiado las rutas de guardado y como se escala la información.

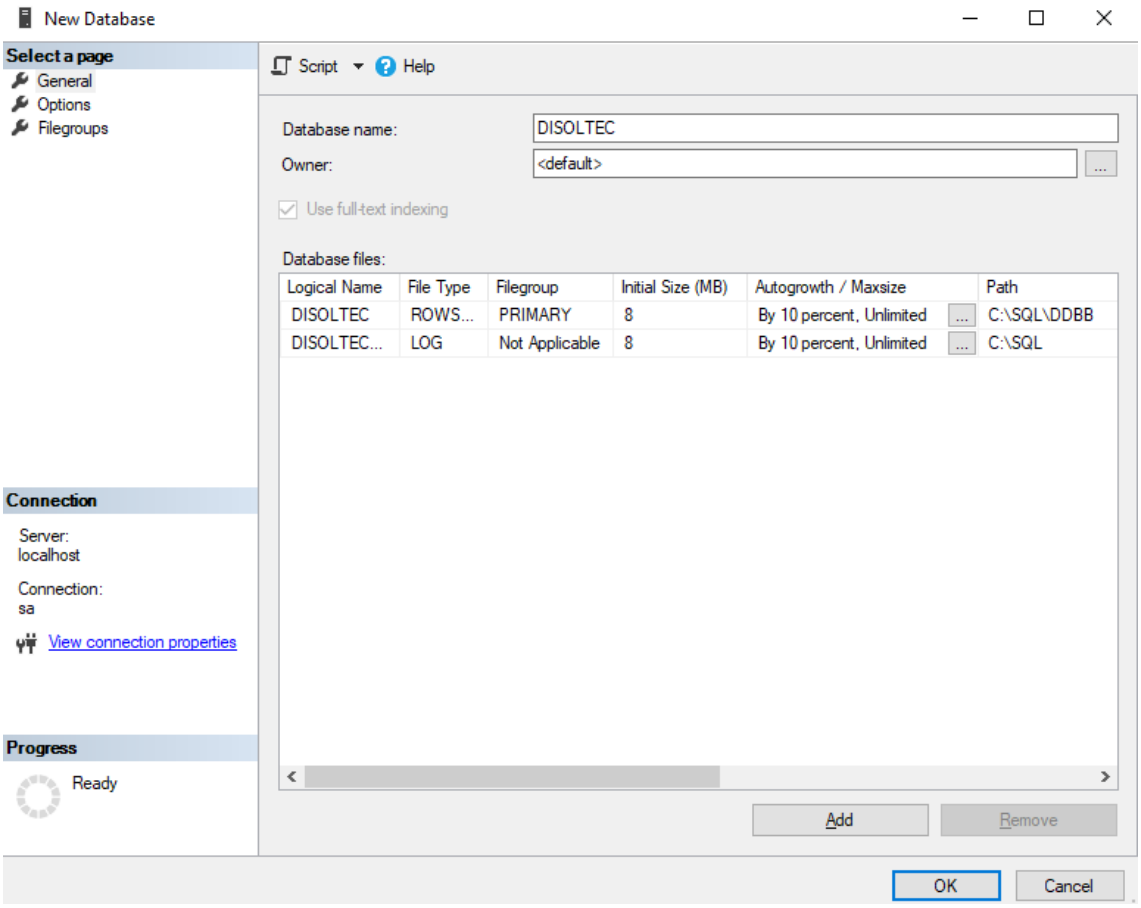



*Figura 27. Configuración de la base de datos*

Como se puede apreciar en la figura anterior, se ha cambiado el crecimiento de la base de un valor fijo a un valor porcentual, esto provoca que las expansiones de la base de datos escalen en relación con la cantidad de información que esta almacena y en consecuencia se produzcan expansiones con menos frecuencia.

Una vez creada y configurada la base de datos sólo queda empezar a llenarla con su estructura, por densidad y porque es un tema que se ha tocado muchas veces durante el grado se ha decidido no documentar la creación de cada una de las tablas descritas anteriormente.

### 6.1.5. Conectar la API con la BBDD

En esta sección se va a exponer el método utilizado en la aplicación para conectar la interfaz de programación de aplicaciones o API con la base de datos SQL Server creada previamente. Actualmente la comunicación entre el cliente y la API se realiza mediante el protocolo HTTP, pero se necesita conectar la API con la base de datos si queremos poder acceder a la información que contiene y poder mostrarla al usuario.

Lo primero que se realiza es crear una clase controlador a la que se ha llamado DataConnection la cual contiene la cadena de conexión a nuestra base de datos y todas las funciones necesarias para realizar consultas sobre esta. Este sería el aspecto que tiene la conexión a la base de datos que hemos creado anteriormente.

```
using System;
using System.Data;

namespace RestAPI_SGADISOLTEC.Recursos
{
    10 referencias
    public class DataConnection
    {
        14 referencias
        public enum Sistemas { SGADISOLTEC }

        // CADENAS DE CONEXION
        const string SGADISOLTECConn = "Server=localhost;Database=SGADISOLTEC;User Id=sa;Pass[redacted]Bdis$;";
        public string DefaultNoData = "NO DATA";

        // FUNCION PARA DEVOLVER LA CADENA DE CONEXION
        1 referencia | 0 excepciones
        private string CadenaConn(Sistemas Sistema)
        {
            string Resultado = "";
            switch (Sistema.ToString())
            {
                case "SGADISOLTEC":
                    Resultado = SGADISOLTECConn;
                    break;
            }
            return Resultado;
        }

        // FUNCION PARA CREAR LA CONEXION CON LA BBDD
        3 referencias | 0 excepciones
        public System.Data.SqlClient.SqlConnection GetConnection(Sistemas Sistema)
        {
            System.Data.SqlClient.SqlConnection cn = new System.Data.SqlClient.SqlConnection();
            cn.ConnectionString = CadenaConn(Sistema);
            cn.Open();

            return cn;
        }
```

*Figura 28. Fragmento de código para realizar la conexión con la base de datos*

Para la conexión de la base de datos, en este caso se ha utilizado el espacio de nombre System.data, el cual brinda funcionalidades y formatos de cadenas de conexión que posibilitan la comunicación con una base de datos.

Analizando la figura 28, podemos observar que primeramente se han creado las variables de conexión que contienen la información de identificación de nuestra base de datos y los datos vacíos por defecto, por otro lado, se aprecia que adicionalmente se crea otra variable de tipo Sistemas por si en el futuro se tuvieran que realizar conexiones a bases de datos distintas.

Se crea conjuntamente al razonamiento anterior, una función llamada CadenaConn que recibe como argumento el sistema que queremos consultar y devuelve la cadena de conexión que corresponde a la base de datos, es interesante recalcar que esta modificación no es necesaria para su funcionamiento, no obstante, se consideró una buena práctica de cara a futuras modificaciones.

Se observa otra función llamada GetConnection que recibe un sistema como argumento, cuyo funcionamiento consiste en crear una conexión SQL utilizando el espacio de nombres, asociarle la cadena de conexión creada previamente y abrir la conexión para realizar todo tipo de operaciones sobre la base de datos.

Una vez conectados a la base de datos podemos crear en la API todo tipo de funciones que nos permitan hacer directamente consultas SQL sobre la base de datos, ejecutar procedimientos y recibir información. Para este proyecto se ha decidido tratar las respuestas de la base de datos como JSON y utilizar estas respuestas para construir tablas y extraer información entre otras funcionalidades.

### 6.1.6. Construcción de la interfaz

Para la creación de la interfaz de nuestra aplicación web se ha utilizado JSON como documento de comunicación entre nuestra interfaz y nuestra base de datos. De esta manera, la interfaz realiza peticiones a la base de datos de información como puede ser los elementos del menú o la información que tienen que mostrar en cada apartado del sistema.

Un buen ejemplo para ver esto en funcionamiento es la construcción del menú. Como se ha explicado en apartados anteriores, el menú se encuentra almacenado en la base de datos y cuando un usuario entra a una página se realiza una petición a la API con una consulta que se ejecuta en la base de datos y que devuelve como respuesta un JSON con todos los elementos que tenemos que cargar. Este es un ejemplo de la respuesta que proporciona la API.


*Figura 29. Fragmento de petición de menú a la API*

Como se aprecia en la figura anterior, el sistema nos devuelve un JSON con toda la información de los apartados que tienen que ser generados en el menú. Para esta aplicación en concreto, Sistema hace referencia a el apartado del que nos estamos refiriendo y toda la lista de módulos que contiene son los subapartados o desplegables que tiene que cargar, si nos fijamos vemos que estos módulos llevan un campo llamado Url que muestra con exactitud la ruta a la que tiene que llevar si el usuario selecciona este subapartado.

Una vez se recibe en el lado del cliente esta información, esta se almacena en una variable de angular y utilizando las propias directivas de angular como ng-repeat para realizar bucles sobre listas, las etiquetas de HTML para realizar la estructura y la barra de navegación que nos ofrece Bootstrap podemos construir el menú. Este es un ejemplo del código para construir el menú a partir del JSON anterior.

```
<nav class="navbar navbar-default navbar-fixed-top">
    <div class="container-fluid">
        <!-- Brand and toggle get grouped for better mobile display -->
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
                <span class="sr-only">MENU</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a href="/home.html"><img class="navbar-brand" src="/imagenes/LogoBlanco.png" alt="SGADISOLTEC" /></a>
            <label style="font-size:20px; margin-top:12px; color:■white; margin-right:20px">SGADISOLTEC</label>
        </div>
        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav">
                <li ng-repeat="sistema in JSONgetMenu" ng-class="sistema.Modulos == '' ? '' : 'dropdown'">
                    <a ng-if="sistema.Url != ''" href="{{sistema.Url}}" ng-class="Aviso>0&&sistema.Sistema=='AVISOS' ? 'aviso' : ''"><span class="{{sistema.Icono}}"></span>
                          {{sistema.Sistema}}
                    </a>
                    <a ng-if="sistema.Url == ''" href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">
                        <span class="{{sistema.Icono}}"></span>  {{sistema.Sistema}} <span class="caret"></span>
                    </a>
                    <ul ng-show="sistema.Modulos != ''" class="dropdown-menu">
                        <li ng-repeat="modulo in sistema.Modulos">
                            <a href="{{modulo.Url}}" title="{{modulo.Descripcion}}"><span class="{{modulo.Icono}}"></span>  {{modulo.Modulo}}</a>
                        </li>
                    </ul>
                </li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
                <li class="dropdown">
                    <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false"><span class="glyphicon glyphicon-user"></span>
                        <span class="caret"></span></a>
                    <ul class="dropdown-menu">
                        <li><a><span class="glyphicon glyphicon-user"></span>   {{UserNombre}}</a></li>
                        <li role="separator" class="divider"></li>
                        <li><a href="/usuario/perfil.html">Perfil</a></li>
                        <li><a href="/usuario/contrasena.html">Contraseña</a></li>
                        <li role="separator" class="divider"></li>
                        <li><a href="javascript:void(0)" ng-click="logout()">Cerrar sesión</a></li>
                    </ul>
                </li>
            </ul>
        </div><!-- /.navbar-collapse -->
    </div><!-- /.container-fluid -->
</nav>
```

*Figura 30. Código para construir el menú*

Utilizando esta misma estrategia se han construido todas las tablas de esta aplicación, utilizando las propias etiquetas de tablas de HTML, JSONs y bucles para crear todas las filas, columnas y botones. En este caso, el JSON que se genera para crear la tabla es muy denso, por tanto, sólo se va a mostrar el resultado final de una tabla generada con este método.


*Figura 31. Ejemplo de tabla Productos*

# 7. Resultados y mejoras

## 7.1. Resultados

Después de haber realizado todo el proceso descrito en los puntos previos, desde el análisis del problema, su entendimiento, proponer una solución y finalmente el desarrollo e implantación la solución, sólo resta mostrar el resultado final de la solución y de esta manera ver la aplicación web en funcionamiento.

Para visualizar el funcionamiento de la aplicación y debido a que mostrar todas las características de la aplicación sería algo muy extenso, se ha decidido hacer una trazabilidad completa de algunos productos, desde que se da entrada en el sistema a estos, hasta que un producto sale del almacén, pasando por su ubicación en el mismo, su reubicación. Por otro lado, se va a mostrar también el formulario que rellenan los empleados para realizar los partes de trabajo y su posterior impresión en formato PDF.

Lo primero que se debe realizar es acceder al sistema, para ello bastará con acceder a la aplicación web mediante el enlace y el puerto que previamente se ha definido en el servidor web. Una vez en la página de la aplicación esta nos pedirá que nos identifiquemos, para ello rellenamos el usuario y la contraseña y pulsamos el botón para identificarse o hacer *login* como se puede ver en la siguiente figura.


*Figura 32. Página de identificación index.html*

Una vez identificado en la aplicación web debemos acceder a la pestaña Entradas desde la barra de navegación, dentro de esta página podremos encontrar una tabla descriptiva con todas las entradas que hay registradas en el momento en el sistema.

El siguiente paso sería añadir una entrada del producto que hayamos comprado o de los productos que esperamos recibir en el almacén, para ello debemos pulsar el botón Insertar entrada que se puede ver enmarcado en rojo en esta figura.


*Figura 33. Fragmento de la tabla entradas*

Después de pulsar el botón de insertar entrada, el sistema nos mostrará una ventana en medio de la pantalla en la que nos pedirá que rellenemos el tipo de producto, el origen de este y la cantidad que son requeridos para insertar una entrada en el sistema. Cuando se hayan rellenado los datos basta con pulsar Sí para dar de alta la entrada.

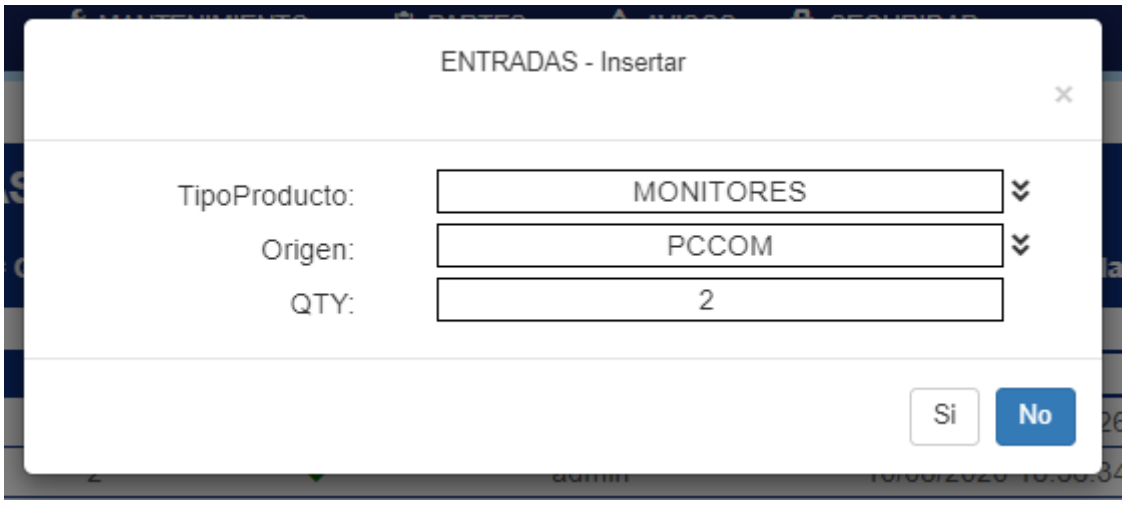



*Figura 34. Ventana para la inserción de una entrada en el sistema*

Una vez insertada la entrada el sistema recargará la tabla y ya se podrá visualizar la nueva entrada en la tabla. En la siguiente figura se muestra la línea de la tabla que ha generado la inserción.

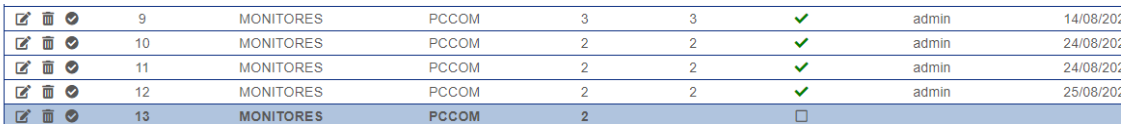



*Figura 35. Líneas actualizadas en la tabla entradas*

Como vemos en la figura 35, la entrada 13 corresponde con la entrada que se ha introducido anteriormente, esta entrada se mantiene en este estado hasta que se reciban los productos en el almacén. Una vez recibidos los productos el operario debe procesar la entrada, para ello basta con pulsar el botón que tiene forma de *tick* en la línea de la entrada. Después de pulsar el botón, el sistema desplegará otra ventana dónde se debe introducir la cantidad recibida para comprobar si es la misma cantidad que se esperaba recibir.

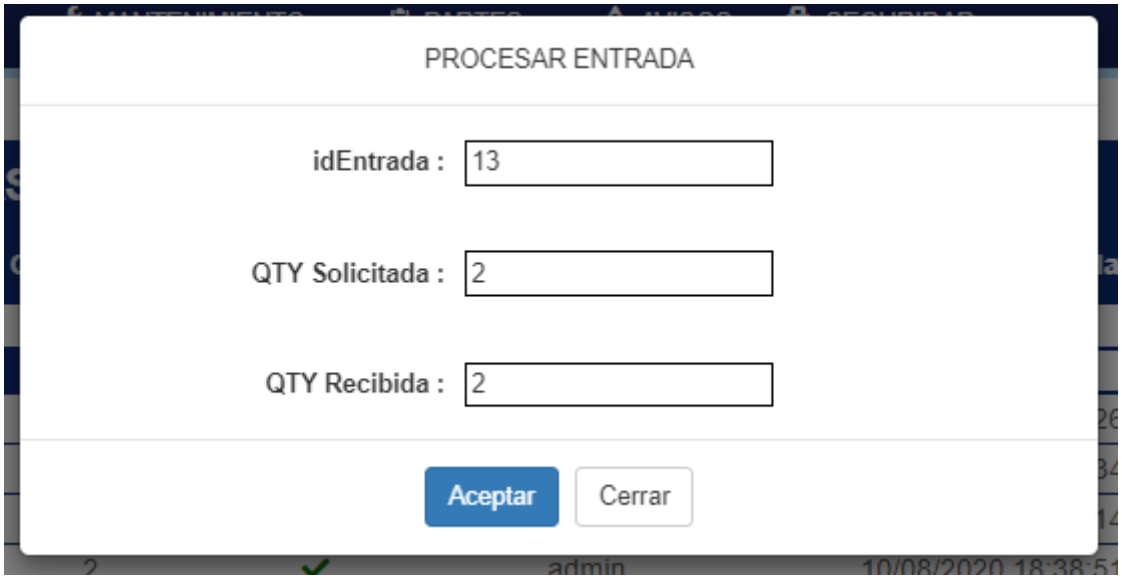



*Figura 36. Ventana para procesar entrada*

Al procesar una entrada automáticamente el sistema crea los productos asociados a la entrada según la cantidad recibida y manda una petición a la impresora para que imprima las etiquetas correspondientes de los productos. En la siguiente figura se puede visualizar el aspecto que tienen estas etiquetas.


*Figura 37. Ejemplo de etiquetas para los productos*

En este punto, el operario debe pegar las etiquetas en los productos para que estos puedan ser o bien leídos utilizando el QR o bien escritos utilizando el identificador de producto. Si en este punto consultamos el estado del producto utilizando la herramienta Consultas, podremos ver que este producto se encuentra en DZ010000 y está activo en el sistema, esto es debido a que la posición DZ010000 es considerada por el sistema la zona de recepción de los productos y por tanto una vez se procesa la entrada, los productos son ubicados automáticamente en esta ubicación.


*Figura 38. Consulta del producto 29 en el sistema*

A partir de aquí el operario puede reubicar el producto a cualquier sitio del almacén que sea elegible, para realizar esta acción debe acceder a la pestaña Reubicar, en esta página se pedirá leer en primer lugar el producto que se quiere reubicar y seguidamente se debe leer la etiqueta de ubicación que se encuentra en la estantería dónde se debe ubicar. Si la información es correcta el sistema cambiará la ubicación del producto por la nueva ubicación y deberá dejar el producto en la nueva ubicación.


*Figura 39. Formato de las etiquetas para las ubicaciones*

El siguiente paso en la trazabilidad del producto se produce cuando se quiere dar salida a uno o más productos del sistema. Para ello, desde la barra de navegación de la aplicación se debe acceder a Salidas, una vez entremos en Salidas el sistema nos muestra una tabla con todas las salidas que se han insertado en el sistema y su correspondiente información. Del mismo modo que las entradas, para insertar una salida debemos pulsar el botón insertar salida, tal y como se aprecia en la siguiente figura.


*Figura 40. Ejemplo de la tabla salidas*

Al pulsar el botón de insertar salida, el sistema despliega una ventana nueva en la que el empleado puede leer o introducir el identificador del producto o productos que quiere dar salida del sistema. De forma dinámica el sistema irá mostrando una lista con los detalles de los productos introducidos, una vez el empleado ha leído todos los productos que quiere pulsa el botón procesar salida como se ve en la figura.

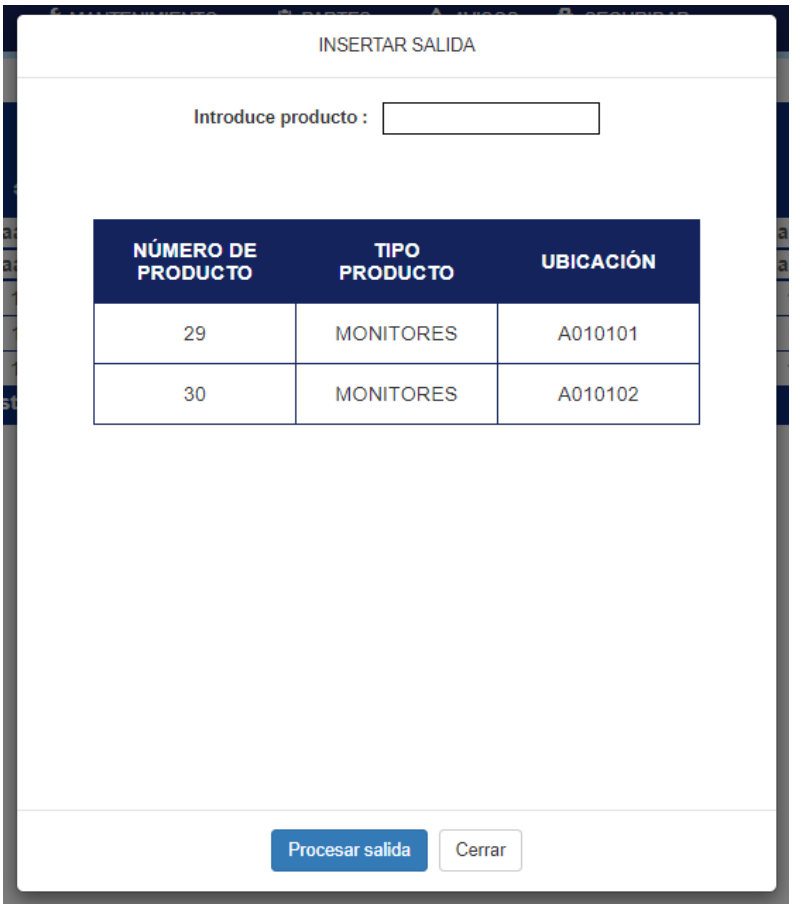



*Figura 41. Ventana para gestionar la salida de productos*

En el momento en el que el empleado procesa la salida, automáticamente el sistema desubica los productos del almacén, libera las ubicaciones y desactiva los productos del sistema, una vez procesada la salida los productos ya se pueden extraer del almacén sin ningún problema. Como se puede ver en la siguiente figura, si en este momento consultamos la situación de uno de los dos productos a los que se les ha dado salida, podremos comprobar que el sistema ha realizado todas las acciones descritas anteriormente.


*Figura 42. Situación del producto después de procesar su salida*

Para utilizar la herramienta para introducir partes de trabajo, se accede a Partes y en el desplegable pulsamos Nuevo Parte. Una vez hecho esto se cargará el formulario que se muestra a continuación, en el cual se tienen que rellenar los campos relativos al parte que queramos registrar. Una vez rellenados los campos se pulsa el botón Registrar, si el formulario es correcto los datos del formulario se vacían, en caso contrario, se muestra un mensaje de error indicando que campo está mal completado.

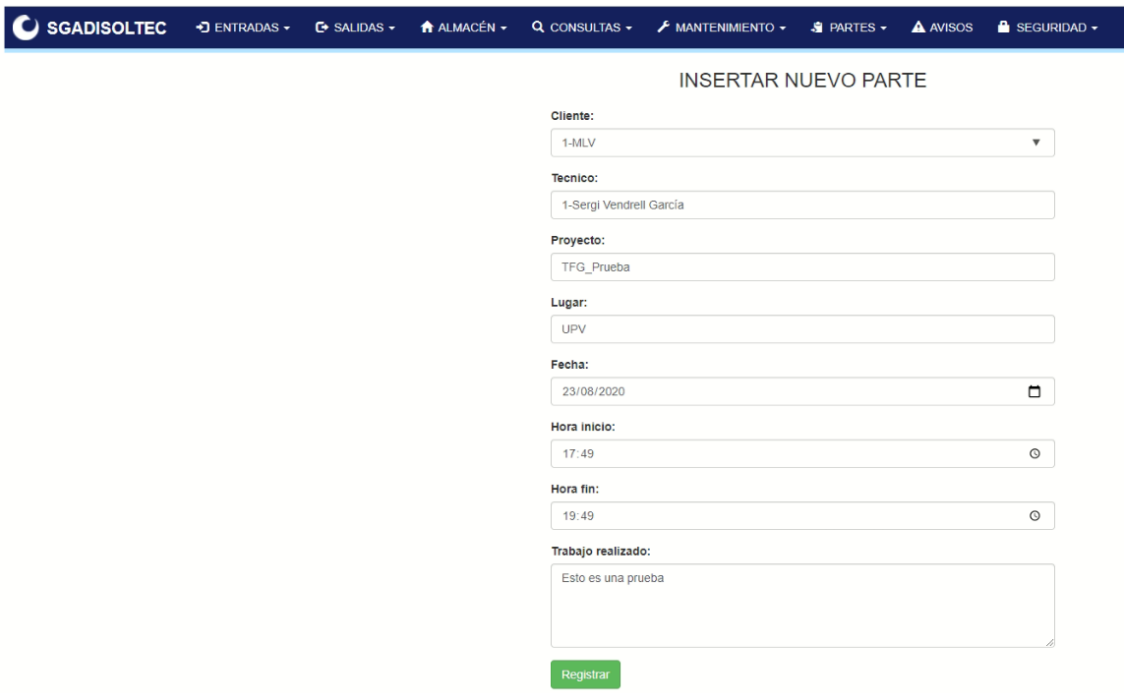



*Figura 43. Formulario para registrar un parte*

Una vez registrado el parte en el sistema, si nos desplazamos a el apartado Partes dentro de Partes, podremos ver que el sistema ha insertado correctamente el parte. En este punto, el parte ya queda registrado en los partes del sistema, no obstante, como Disoltec comunicó que en ocasiones era necesario la impresión del mismo parte, se habilitó un

botón con forma de impresora que imprime directamente el parte PDF en el sistema. Este botón se puede ver en la siguiente Figura dentro de un círculo rojo.

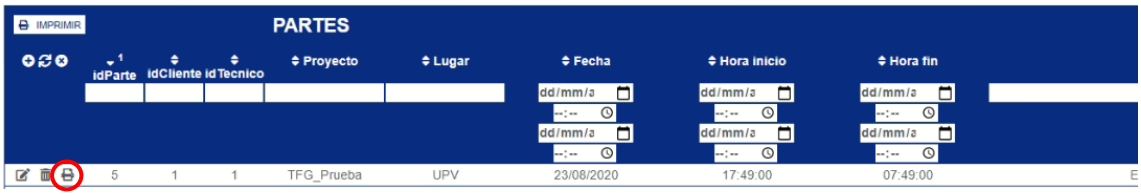
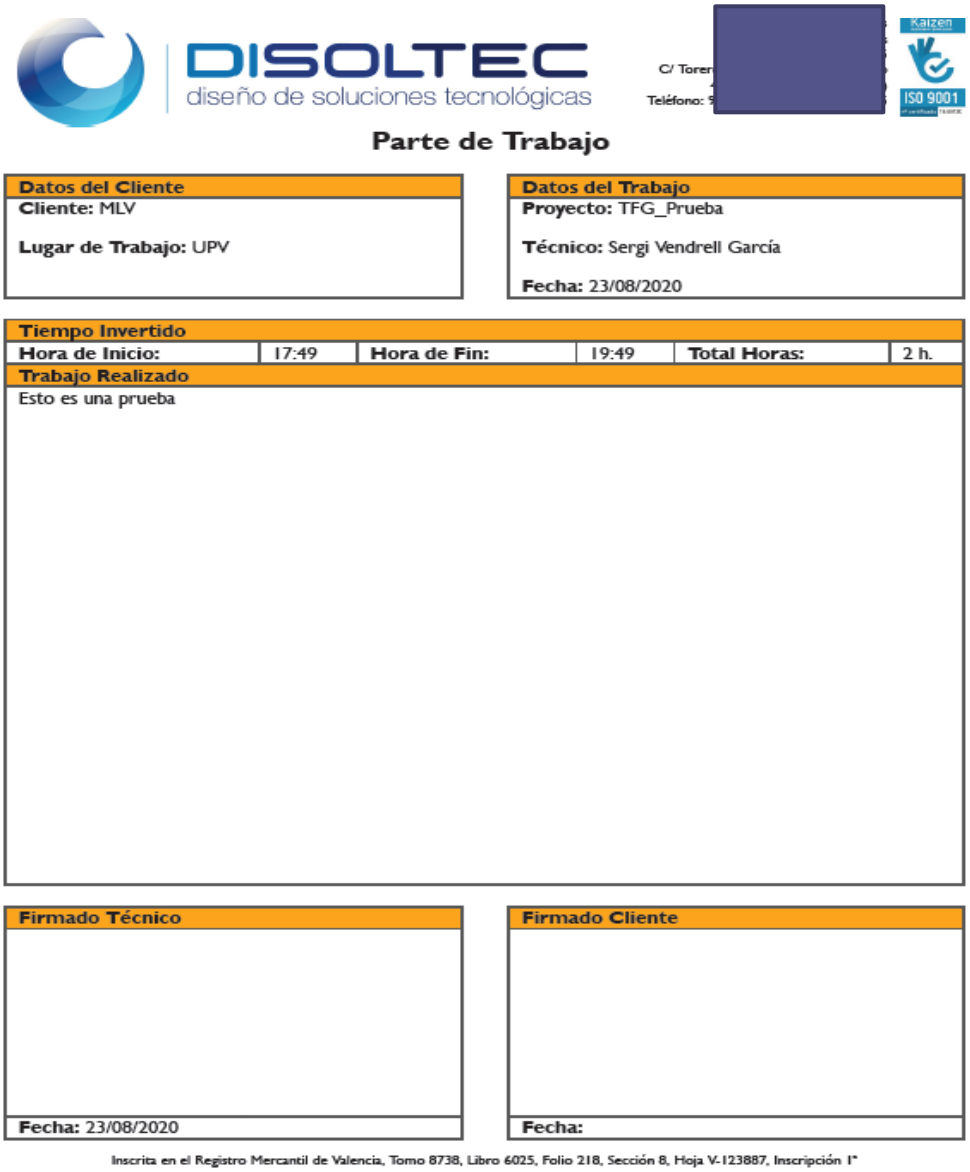



*Figura 44. Visualización de la tabla de Partes*

Justo después de pulsar el botón que se muestra en la anterior Figura, se despliega otra página HTML que genera un documento PDF descargable para imprimir en cualquier impresora o ser almacenado en el cualquier ordenador de forma local, el parte tiene el aspecto que muestra la siguiente Figura, se han censurado algunas partes para proteger los datos de Disoltec.


DISOLTEC
diseño de soluciones tecnológicas

C/ Torer
Teléfono:

Kaizen
ISO 9001

**Parte de Trabajo**

| Datos del Cliente | Datos del Trabajo |
|---|---|
| **Cliente:** MLV | **Proyecto:** TFG_Prueba |
| **Lugar de Trabajo:** UPV | **Técnico:** Sergi Vendrell García |
| | **Fecha:** 23/08/2020 |

| Tiempo Invertido | | | | | |
|---|---|---|---|---|---|
| **Hora de Inicio:** | 17:49 | **Hora de Fin:** | 19:49 | **Total Horas:** | 2 h. |

**Trabajo Realizado**

Esto es una prueba

| Firmado Técnico | Firmado Cliente |
|---|---|
| **Fecha:** 23/08/2020 | **Fecha:** |

Inscrita en el Registro Mercantil de Valencia, Tomo 8738, Libro 6025, Folio 218, Sección 8, Hoja V-123887, Inscripción 1ª

*Figura 45. Ejemplo de impresión de un parte de trabajo*

## 7.2. Mejoras

Llegados a este punto del proyecto, se han detectado diferentes características o herramientas que finalmente no han formado parte del proyecto ya sea por el límite de extensión que supone un trabajo de fin de grado o debido a que su implementación requeriría un estudio más profundo. Este apartado tiene la función de recopilar y dejar constancia de todos los aspectos en los que se podrían centrar las futuras implementaciones y correcciones sobre este trabajo de fin de grado y la aplicación web resultante.

Estos son algunos de los aspectos se han detectado y resultaría interesante explorar en futuras implementaciones, con el único fin de mejorar la aplicación en términos generales y dotarla de mayor autonomía logística.

- Restricciones: aunque el almacén que se plantea en este trabajo de fin de grado sigue una estrategia caótica, la mayoría de los almacenes incluyen restricciones que pueden ayudar en la eficiencia de este. Por ejemplo, si suponemos que tenemos un tipo de producto A y un tipo de producto B y ambos deben ser extraídos del almacén de manera conjunta, podría ser coherente estudiar la posibilidad de almacenar siempre un producto B en las celdas contiguas de un producto A.
- Rutas de *picking:* si queremos que un almacén sea eficiente y escalable, las rutas de *picking* son algo esencial, una mala ruta de recogida de productos puede resultar en un almacén extremadamente ineficiente y dotarlo de poca escalabilidad. Este aspecto en cuestión es muy interesante ya que, habría que estudiar el mejor algoritmo de rutas para la disposición de nuestro almacén.
- *Layout o* disposición del almacén: como hemos comentado en el punto anterior, la disposición de las estanterías y ubicaciones dentro del almacén importa, es por eso por lo que entra en la lista de aspectos a mejorar en futuras mejoras de este.
- Mejora de etiquetas: para este gestor de almacén se han realizado unas etiquetas muy sencillas utilizando códigos QR que en términos generales son más que suficientes, no obstante, existe mucho margen de mejora en este ámbito, desde ajustar el tamaño de las etiquetas para abaratar costes hasta añadir más información de trazabilidad en las mismas.
- Radiofrecuencia para stock: la radiofrecuencia es una tecnología que cada vez se está aprovechando más en la logística, dotar los productos de etiquetas con radiofrecuencia podría simplificar enormemente la complejidad del almacén, quitando carga a los operarios y añadiendo procesos automáticos que gestionen autónomamente el almacén. Un buen ejemplo, sería utilizar las etiquetas con radiofrecuencia para realizar inventarios automáticos detectando la cantidad de etiquetas que existen en el almacén mediante antenas, otro ejemplo podría ser seguir la trazabilidad de los productos por el almacén del mismo modo.

# 8. Conclusiones

Llegados a este capítulo, se van a exponer las conclusiones percibidas a lo largo de la realización de este trabajo de fin de grado. Primeramente, se van a contrastar los resultados obtenidos con los objetivos planteados al inicio del trabajo para poder comprobar en que grado han sido satisfechos. Por último, se van a exponer las conclusiones personales recogidos durante la realización de este trabajo de fin de grado.

Respecto a los objetivos propuestos al inicio del trabajo sobre la aplicación web a desarrollar, contrastando con el cliente obtenemos que:

- Se ha conseguido optimizar el espacio en las instalaciones, después de implementar la aplicación se ha aprovechado aproximadamente un 30% del almacén que ahora puede ser destinado para otros fines.
- Percibida una agilización en el tiempo que los empleados pasan en el almacén, con el sistema actual se accede al almacén el tiempo justo y no se pierde tiempo buscando dónde están los productos.
- Los usuarios que utilizan la aplicación dicen estar muy satisfechos con la aplicación y está siendo utilizada en la actualidad.
- Se ha conseguido dotar a la aplicación de versatilidad ya que la aplicación puede ser utilizada tanto con un ordenador o como con un dispositivo móvil con lector QR indistintamente.

Haciendo una aproximación utilizando las horas empleadas, alrededor de 500 y el material que se utiliza en el almacén, entre 2000€ - 4000€ podríamos estimar que el precio de la inversión total podría rondar entre 8000–12000€ dependiendo del grado de experiencia del equipo de desarrolladores y suponiendo que tendremos que realizar cierta inversión en material de etiquetaje y lectoras.

Con toda esta información, concluimos que se han cumplido la mayoría de los objetivos propuestos al inicio del trabajo y la solución desarrollada es amortizable teniendo en cuenta la eficiencia y el espacio que se ha conseguido ganar en las instalaciones. Contrastando que el precio del metro cuadrado en Catarroja es de aproximadamente 1000€ y se han conseguido aprovechar más o menos 50m2, observamos que, sólo con la ganancia de espacio la realización de esta aplicación resultaría muy rentable. (Trovimap, 2020)

Para finalizar vamos a expresar un poco las conclusiones personales que se han extraído durante este trabajo de fin de grado. Ha resultado muy interesante encontrarse en la tesitura de realizar una especificación desde cero con un cliente y experimentar de esta manera cómo se realizan los proyectos fuera de un ámbito académico. Asimismo, trabajar con herramientas nuevas y aprender sobre ellas ha sido una experiencia muy cercana al mundo laboral e indudablemente enriquecedora.

Por otro lado, durante la realización del proyecto se ha revindicado la importancia de realizar una buena especificación ya que todos los cabos sueltos que se puedan quedar en la fase de especificación de requisitos pueden suponer una gran pérdida de esfuerzo y dinero en la parte de desarrollo. Con todo esto, la realización de este proyecto ha resultado ser de un enorme ejercicio de aprendizaje en el que se han visto materializados la mayoría de los conocimientos adquiridos en el grado.

# 9. Bibliografía


AngularJS (no date) *Angular JS*. Available at: https://docs.angularjs.org/guide/introduction.

Apache (1995) *Apache*. Available at: https://httpd.apache.org/.

Brooks, D. R. (2007) *An Introduction to HTML and JavaScript [electronic resource] : for Scientists and Engineers*. 1st ed. 2007.

Datadec (2020) *No Title*. Available at: https://www.mecalux.es/.

Dauzon, S., Bendoraitis, A. and Ravindran, A. (2016) *Django*. Packt Publishing.

DB-Engines (2020) *Ranking DBMS*. Available at: https://db-engines.com/en/ranking/relational+dbms.

DELL (2020) *DELL PowerEdge R420*. Available at: https://www.dell.com/es-es/work/shop/povw/poweredge-r420.

Disoltec (2007a) *Conócenos*. Available at: http://www.disoltec.es/conocenos/.

Disoltec (2007b) *Soluciones*. Available at: http://www.disoltec.es/soluciones/.

Dongil Sánchez, J. A. (2018) *Por qué elegir VueJS: 5 razones para considerarlo nuestro próximo framework de referencia*, *11 de Febrero*. Available at: https://www.genbeta.com/desarrollo/por-que-elegir-vuejs-5-razones-para-considerarlo-nuestro-proximo-framework-de-referencia.

Duckett, J. (2011) *HTML and CSS*. 1. Aufl. US: Wiley.

Durán Toro, A. and Bernárdez Jiménez, B. (2002) 'Metodología para la Elicitación de Requisitos de Sistemas Software (versión 2.3)'. Universidad de Sevilla, p. 82. Available at: https://goo.gl/rhV8eV.

EDUCBA (no date) *Advantages of HTML*. Available at: https://www.educba.com/advantages-of-html/.

Hartl, M. (2012) *Ruby on Rails Tutorial, Learn Web Development with Rails*. 4th edn. Addison-Wesley Professional.

INCIBE (2019) 'Qué es una DMZ y cómo te puede ayudar a proteger tu empresa'. Available at: https://www.incibe.es/protege-tu-empresa/blog/dmz-y-te-puede-ayudar-proteger-tu-empresa.

Khan, A. (no date) *Microsoft IIS 10.0 Cookbook*. 1st edn. GB: Packt Publishing.

McFarland, D. S. (2008) *JavaScript : the missing manual*. First edition. (Missing manual).

Mecalux (2020) *No Title*. Available at: https://www.mecalux.es/.

Nginx (2004) *Nginx*. Available at: https://nginx.org/en/.

OBS Business School (2020) *SGA: qué es y qué ventajas supone para la empresa*.

OMG (2006) *BPMN*. Available at: https://www.omg.org/bpmn/.

Patel, P. R. (no date) 'Existence of Dependency-Based Attacks in NodeJS Environment'. Iowa State University Digital Repository.

Petković, D. (2017) *Microsoft SQL Server 2016 : a beginner's guide*. Sixth edition.

*PR Newswire* (2007) 'LiteSpeed Technologies Releases High-Performance Apache Replacement Web Server'.

QLOUDEA (2018) *Armario rack*. Available at: https://qloudea.com/blog/armario-rack-mural/.

Ragupathi, M. T. S., De Sanctis, V. and Singleton, J. (2017) *ASP.NET Core*. 1st ed. PACKT Publishing.

Sahu, P. (no date) *Advantages and Disadvantages of CSS Everyone Should Know*. Available at: https://www.motocms.com/blog/en/advantages-and-disadvantages-of-css/.

Santana Roldan, C. (2018) *ReactJS cookbook*. Packt Publishing.

Schwarzmüller, M. (2019) *React vs. Angular vs. Vue: Which is the Best JavaScript Framework?*, *Agosto*. Available at: https://blog.udemy.com/react-js-vs-angular-vs-vue-js-which-is-the-best-javascript-framework/?gclid=Cj0KCQjw7qn1BRDqARIsAKMbHDb55MWVZNP3BmXPMW_vZ8wUvNO8ILZPwBcw31lhdLMrIaxNGVLoVZcaAiQ7EALw_wcB&matchtype=b&utm_campaign=DSA_Catchall_la.EN_cc.ROW&utm_content=.

SCM Logística (2020) *No Title*. Available at: https://www.scmlogistica.es.

Shakuntala Gupta Edward and Navin Sabharwal (2015) *Practical MongoDB*. 1st edn. Berkeley, CA: Apress (Expert's voice in open source). doi: 10.1007/978-1-4842-0647-8.

SoapUI (no date) *SOAP vs REST 101: Understand The Differences*. Available at: https://www.soapui.org/learn/api/soap-vs-rest-api/.

solid IT (2020) *DB-Engines*. Available at: https://db-engines.com/en/.

Stobart, S. (2004) 'PHP and MySQL manual : simple, yet powerful web programming'. Edited by M. Vassileiou. London [etc.]: Springer (Springer professional computing).

Tejero, J. J. A. (2008) *Almacenes: Análisis, diseño y organización*. Edited by ESIC Editorial.

TicoStyle (2009) *Armario servidores tipo blade*. Available at: https://ticostyle.wordpress.com/2009/09/30/servidores-blades/.

Trovimap (2020) *Histórico de precios en el mercado residencial de Catarroja*. Available at: https://www.trovimap.com/precio-vivienda/valencia-valencia/catarroja.

W3Techs (2020) *Estadísticas de uso de servidores web*. Available at: https://w3techs.com/technologies/overview/web_server.

# Anexo

## A Documentos de especificación de requisitos

### A.1 Requisitos de información

| **IRQ-01** | **Información sobre entradas** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas de los productos | |
| **Requisitos asociados** | • UC-01 Insertar entradas<br>• UC-17 Anular entradas<br>• UC-13 Ubicar producto<br>• UC-14 Consultar entradas<br>• UC-16 Procesar entradas | |
| **Descripción** | *El sistema debe almacenar la información relativa a las entradas que se producen en el almacén, en este caso debe almacenar estos datos:* | |
| **Datos específicos** | • En qué almacén se encuentra<br>• En qué ubicación se encuentra<br>• La cantidad de productos asociada a la entrada<br>• Si está procesada<br>• Quién la ha procesado<br>• Si está anulada<br>• Quién la ha anulado<br>• Cuando fue introducida | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| | 1.5 años | 4 años |
| **Ocurrencias simult.** | **Medio** | **Máximo** |
| | 10 | 20 |
| **Estabilidad** | Alta | |
| **Comentarios** | *Se han introducido los requisitos UC-17, UC-20, UC-21 debido a que el sistema una vez procesa las entradas ubica automáticamente los productos y este también puede ser anulado.* | |

| **CRQ-01** | **Relación entre entradas y ubicaciones** |
|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas de los productos |
| **Requisitos asociados** | • IRQ-01 Información sobre entradas<br>• UC-16 Procesar entradas<br>• UC-13 Ubicar producto |
| **Descripción** | Los datos almacenados en el sistema referentes a las entradas deben cumplir siempre que: *cuando una entrada se encuentra procesada debe ubicarse inicialmente en la ubicación de entrada para posteriormente ser ubicada dónde recomiende el almacén.* |
| **Estabilidad** | Alta |
| **Comentarios** | ninguno |

| **IRQ-02** | **Información sobre salidas** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-02 Gestionar las salidas de los productos | |
| **Requisitos asociados** | • UC-02 Insertar y procesar salidas<br>• UC-15 Consultar salidas<br>• UC-12 Consultar los avisos del almacén | |
| **Descripción** | *El sistema debe almacenar la información relativa a las salidas que se producen en el almacén, en este caso debe almacenar estos datos:* | |
| **Datos específicos** | • Que producto o productos salen del almacén<br>• La ubicación de los productos<br>• La cantidad de productos que salen<br>• Si está procesada<br>• Quién la ha procesado<br>• Si está anulada<br>• Quién la ha anulado<br>• Cuándo fue introducida | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| | 1.5 años | 4 años |
| **Ocurrencias simult.** | **Medio** | **Máximo** |
| | 10 | 20 |
| **Estabilidad** | Alta | |
| **Comentarios** | *Se incluye el requisito UC-15 ya que leerá información sobre salidas para mostrar avisos o notificaciones.* | |

| CRQ-02 | Relación entre salidas y ubicaciones |
|---|---|
| **Objetivos asociados** | • OBJ-02 Gestionar las salidas de los productos |
| **Requisitos asociados** | • IRQ-02 Información sobre las salidas |
| **Descripción** | Los datos almacenados en el sistema referentes a las entradas deben cumplir siempre que: *siempre que se de salida a algún producto debe dejar libre la ubicación dónde se encontraba y actualizar el stock del producto.* |
| **Estabilidad** | Alta |
| **Comentarios** | Sería conveniente estudiar también la posibilidad de que el propio sistema fuese capaz de avisar a los usuarios de cuándo es necesario comprar nuevos productos para mantener stock. |

| IRQ-03 | Información sobre trazabilidad de productos | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad de los productos | |
| **Requisitos asociados** | • UC-03 Consultar el estado del almacén<br>• UC-04 Reubicar producto<br>• UC-05 Insertar productos<br>• UC-06 Consultar productos<br>• UC-07 Consultar los productos que se han ubicado<br>• UC-08 Consultar ubicación<br>• UC-11 Consultar los logs<br>• UC-14 Consultar entradas<br>• UC-15 Consultar salidas<br>• UC-16 Procesar entradas<br>• UC-17 Anular entradas | |
| **Descripción** | *El sistema debe almacenar la información relativa a la trazabilidad de productos que se producen en el almacén, en este caso debe almacenar estos datos:* | |
| **Datos específicos** | • Dónde se encuentra el producto en todo momento<br>• Comprobar si se le ha dado entrada a los productos<br>• Comprobar si se le ha dado salida a los productos<br>• Consultar los registros para ver las acciones que se le han realizado a los productos.<br>• Información acerca del motivo por el que se le ha realizado alguna acción sobre el producto. | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| | 1.5 años | 4 años |
| **Ocurrencias simult.** | **Medio** | **Máximo** |
| | 10 | 20 |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **CRQ-03** | **Relación entre productos, restricciones y ubicaciones** |
|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad de los productos |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad de productos |
| **Descripción** | El sistema debe tener en cuenta para la trazabilidad de los productos que: *pueden existir restricciones dentro del almacén que deben ser interpretadas para la ubicación y trazabilidad de los productos.* |
| **Estabilidad** | Alta |
| **Comentarios** | Se deja constancia de un ejemplo de restricción que podría existir en el almacén: *los cables de red tienen una restricción por la cual no pueden ser ubicados en las estanterías A y C.* |

| **IRQ-04** | **Información de stock** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • UC-16 Procesar entradas<br>• UC-02 Insertar y procesar salidas<br>• UC-03 Consultar el estado del almacén<br>• UC-05 Insertar productos<br>• UC-06 Consultar productos<br>• UC-12 Consultar avisos del almacén<br>• UC-17 Anular entradas<br>• UC-09 Insertar tipos de ubicaciones<br>• UC-10 Consultar tipos de ubicaciones | |
| **Descripción** | *El sistema debe almacenar la información relativa al stock que tiene el almacén en este caso debe almacenar estos datos:* | |
| **Datos específicos** | • Cantidad de espacio libre<br>• Cantidad de productos ubicados<br>• Tipos de ubicaciones<br>• Cantidad de productos que tienen entradas asociadas<br>• Cantidad de productos que tienen salidas asociadas<br>• Cantidades mínimas | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| | 1.5 años | 4 años |
| **Ocurrencias simult.** | **Medio** | **Máximo** |
| | 10 | 20 |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| CRQ-04 | Relación entre stock y mínimos |
|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock |
| **Requisitos asociados** | • IRQ-04 Información del stock |
| **Descripción** | El sistema debe tener en cuenta para la gestión del stock que: *debido a que la empresa debe tener siempre en el almacén ciertos productos, existen unos mínimos que deben ser cubiertos por la gestión del stock* |
| **Estabilidad** | Alta |
| **Comentarios** | - |

| IRQ-05 | Información sobre partes de trabajo | |
|---|---|---|
| **Objetivos asociados** | • OBJ-05 Gestión de partes de trabajo | |
| **Requisitos asociados** | • UC-28 Insertar parte de trabajo<br>• UC-29 Modificar parte de trabajo<br>• UC-30 Eliminar parte de trabajo | |
| **Descripción** | *El sistema debe almacenar la información relativa a los partes de trabajo que se generen en el sistema, en este caso debe almacenar:* | |
| **Datos específicos** | • Quién realiza el parte<br>• Para que empresa es el parte<br>• Para que proyecto es el parte<br>• Qué día se realizó<br>• A qué hora empezó el trabajo<br>• A qué hora finalizó el trabajo | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| | 1.5 años | 4 años |
| **Ocurrencias simult.** | **Medio** | **Máximo** |
| | 10 | 20 |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

## A.2 Casos de uso del sistema

| | | |
|---|---|---|
| **UC-01** | **Insertar entrada** | |
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas de productos | |
| **Requisitos asociados** | • IRQ-01 Información sobre entradas | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente insertar una entrada en el almacén.* | |
| **Precondición** | El usuario realiza una compra o tiene conocimiento de que se van a recibir productos. | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de entradas |
| | 2 | El empleado pulsa el botón de insertar entrada que se muestra en la página entradas |
| | 3 | El sistema muestra una ventana en la que el empleado debe introducir los datos necesarios. |
| | 4 | El empleado rellena los datos y confirma la inserción |
| | 5 | El sistema introduce los datos en el almacén |
| **Postcondición** | Se ha insertado una entrada nueva | |
| **Excepciones** | **Paso** | **Acción** |
| | 4 | Si la entrada ya existe en el sistema se le comunica al empleado |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 5 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/dia | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-02 | Insertar y procesar salidas | |
|---|---|---|
| **Objetivos asociados** | • OBJ-02 Gestionar las salidas de productos | |
| **Requisitos asociados** | • IRQ-02 Información sobre salidas | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente insertar una salida en el almacén.* | |
| **Precondición** | El empleado desea insertar una salida nueva en el almacén, esta salida tiene que estar vinculada con uno o más productos dados de alta previamente. | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de salidas |
| | 2 | El empleado pulsa el botón de insertar salida que se muestra en la página salidas |
| | 3 | El sistema muestra una ventana en la que el empleado debe leer todos los productos a los que quiere dar salida del almacén |
| | 4 | El empleado rellena los datos y confirma el procesamiento de la salida |
| | 5 | El sistema introduce los datos en el almacén |
| **Postcondición** | Se ha insertado y procesado una salida existente | |
| **Excepciones** | **Paso** | **Acción** |
| | 4 | Si ha leído algún producto no válido se le notifica al cliente |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 5 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/dia | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-03 | Consultar el estado del almacén | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad del producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente consultar el estado en tiempo real del almacén.* | |
| **Precondición** | El empleado desea tener una idea general del estado actual del almacén. | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de almacén |
| | 2 | El sistema carga una tabla que muestra el estado de almacén en el momento que el empleado lo está visitando y su almacenamiento. |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 5 segundos |
| **Frecuencia** | Aprox. 10 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-04** | **Reubicar productos** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad del producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente cambiar la ubicación de un producto* | |
| **Precondición** | El empleado desea reubicar un producto en otra ubicación del almacén indistintamente el motivo. | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de reubicar |
| | 2 | El empleado introduce o lee el identificador del producto que quiere reubicar |
| | 3 | El sistema muestra un detalle del producto introducido y deja un campo para introducir la nueva ubicación en caso de hacerlo manualmente. |
| | 4 | El empleado rellena los datos o lee la ubicación y pulsa el botón de reubicar |
| | 5 | El sistema reubica el producto en el almacén |
| **Postcondición** | Se ha reubicado el producto en otra ubicación distinta | |
| **Excepciones** | **Paso** | **Acción** |
| | 3 | Si introduce un identificador incorrecto o invalido el sistema lo notifica al empleado |
| | 4 | Si introduce datos inconsistentes o incorrectos el sistema lo notificará al empleado |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 5 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-05 | Insertar productos | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *después de que un usuario inserte una entrada en el almacén y la procese*. | |
| **Precondición** | El usuario tiene que haber insertado y procesado previamente una entrada en el almacén | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado introduce con éxito una entrada |
| | 2 | El empleado procesa la entrada |
| | 3 | En el momento de procesar la entrada, el sistema detecta la cantidad de productos y el tipo de productos que contiene la entrada y registra los productos asociados |
| **Postcondición** | Se han insertado productos nuevos al procesar una entrada existente | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 3 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/dia | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-06 | Consultar productos | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad del producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente consultar el estado de un producto*. | |
| **Precondición** | El empleado desea tener una idea general del estado de un producto que se encuentra actualmente en el almacén. | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de consultar producto y lee o introduce un identificador de producto |
| | 2 | El sistema carga una tabla que muestra el estado del producto en el momento que el empleado lo está visitando y su ubicación si está ubicado. |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 2 segundos |
| **Frecuencia** | Aprox. 10 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-07** | **Consultar los productos que se han ubicado** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad del producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente visualizar los productos que se han ubicado en el almacén* | |
| **Precondición** | El empleado desea tener una idea general de los productos que han sido ubicados | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de consultar productos ubicados |
| | 2 | El sistema carga una tabla que muestra una lista de todos los productos que han sido ubicados en el almacén. |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-08** | **Consultar ubicación** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente visualizar el estado de una ubicación* | |
| **Precondición** | El empleado desea tener una idea general del estado de una ubicación, espacio libre, productos que hay en ella | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de consultar ubicación |
| | 2 | El sistema carga una tabla que muestra un campo para introducir un identificador de una ubicación |
| | 3 | El empleado lee o introduce el identificador y pulsa el botón de consultar |
| | 4 | El sistema muestra una tabla que muestra información detallada de la ubicación, espacio libre productos que hay en esta |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | 3 | Si el empleado introduce un identificador no válido el sistema se lo notifica |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 4 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-09 | Insertar tipos de ubicaciones | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente insertar un tipo de ubicación en el sistema*. | |
| **Precondición** | El usuario desea introducir un tipo de ubicación sin ser relevante el motivo | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de tipos de ubicaciones |
| | 2 | El sistema carga la tabla con los tipos de ubicaciones existentes en el sistema |
| | 3 | El empleado pulsa el botón de insertar tipo de ubicación |
| | 4 | El sistema despliega una ventana en la que el empleado tiene que introducir los valores requeridos |
| | 5 | El empleado cumplimenta estos datos i pulsa el botón de insertar |
| | 6 | El sistema inserta la información en el sistema |
| **Postcondición** | Se ha insertado un tipo de ubicación nueva | |
| **Excepciones** | **Paso** | **Acción** |
| | 5 | Si el empleado introduce algún dato incorrecto o inconsistente el sistema se lo notifica |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 6 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-10 | Consultar tipos de ubicaciones | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente visualizar los tipos de ubicaciones existentes en el sistema* | |
| **Precondición** | El empleado desea visualizar los tipos de ubicaciones que existen en el sistema | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de mantenimiento/ tipos de ubicaciones |
| | 2 | El sistema carga una tabla que muestra los tipos de ubicaciones del sistema |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-11** | **Consultar los logs** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Trazabilidad del producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente visualizar los registros del almacén* | |
| **Precondición** | El empleado desea visualizar las modificaciones, ejecuciones, ubicaciones o cambios que se han realizado en el almacén | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de logs |
| | 2 | El sistema carga una tabla que muestra los registros de las modificaciones del sistema |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-12** | **Consultar avisos del almacén** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente consultar los avisos del almacén* | |
| **Precondición** | El empleado desea visualizar las recomendaciones y avisos del almacén | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de avisos |
| | 2 | El sistema carga una tabla que muestra los avisos que ha generado el almacén basándose en los mínimos de cada tipo de producto |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 3 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-13 | Ubicar producto | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad del producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *después de que se procese una entrada y se inserten los productos*. | |
| **Precondición** | Los productos que se quieren ubicar tienen que estar insertados | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado introduce con éxito una entrada |
| | 2 | El empleado procesa la entrada |
| | 3 | En el momento de procesar la entrada, el sistema detecta la cantidad de productos que contiene la entrada y registra los productos asociados |
| | 4 | Una vez registrados los productos, el sistema los almacena automáticamente en una ubicación profesional |
| **Postcondición** | Se han ubicado los productos que se han generado | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 4 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/dia | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| UC-14 | Consultar entradas | |
|---|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas de productos | |
| **Requisitos asociados** | • IRQ-01 Información sobre entradas | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente consultar las entradas en el sistema*. | |
| **Precondición** | El empleado desea tener una idea general de las entradas. | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de entradas |
| | 2 | El sistema carga una tabla que muestra información relacionada con el estado de las entradas existentes en ese momento en el sistema |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 2 segundos |
| **Frecuencia** | Aprox. 10 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-15** | **Consultar salidas** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-02 Gestionar las salidas de productos | |
| **Requisitos asociados** | • IRQ-02 Información sobre salidas | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente consultar las salidas en el sistema.* | |
| **Precondición** | El empleado desea tener una idea general de las salidas. | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de salidas |
| | 2 | El sistema carga una tabla que muestra información relacionada con el estado de las salidas existentes en ese momento en el sistema |
| **Postcondición** | ninguno | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 2 segundos |
| **Frecuencia** | Aprox. 10 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-16** | **Procesar entradas** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas | |
| **Requisitos asociados** | • IRQ-01 Información sobre entradas | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente procesar una entrada del almacén* | |
| **Precondición** | Para procesar una entrada esta debe existir con anterioridad | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de entradas |
| | 2 | El sistema carga la tabla con las entradas existentes en el sistema |
| | 3 | El empleado pulsa el botón de procesar entrada |
| | 4 | El sistema despliega una ventana en la que el empleado tiene que introducir la cantidad de productos recibidos |
| | 5 | El empleado confirma el procesamiento de la entrada |
| | 6 | El sistema inserta la información en el sistema y realiza los casos de uso UC-05 y UC-13 |
| **Postcondición** | Se ha procesado una entrada | |
| **Excepciones** | **Paso** | **Acción** |
| | 5 | Si el empleado introduce algún dato incorrecto o inconsistente el sistema se lo notifica |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 6 | 3 segundos |
| **Frecuencia** | Aprox. 2 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-17** | **Anular entradas** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-01 Gestionar las entradas | |
| **Requisitos asociados** | • IRQ-01 Información sobre entradas | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente anular una entrada del almacén* | |
| **Precondición** | Para anular una entrada esta debe existir con anterioridad y debe de estar sin procesar | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de entradas |
| | 2 | El sistema carga la tabla con las entradas existentes en el sistema |
| | 3 | El empleado pulsa el botón de modificar entrada |
| | 4 | El sistema despliega una ventana en la que el empleado tiene que introducir los valores modificables |
| | 5 | El empleado marca la entrada como anulada i pulsa el botón de modificar |
| | 6 | El sistema inserta la información en el sistema y realiza los cambios si fuesen necesarios |
| **Postcondición** | Se ha anulado una entrada | |
| **Excepciones** | **Paso** | **Acción** |
| | 5 | Si el empleado introduce algún dato incorrecto o inconsistente el sistema se lo notifica |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 6 | 3 segundos |
| **Frecuencia** | Aprox. 2 veces/semana | |
| **Estabilidad** | Alta | |
| **Comentarios** | Si el empleado anula una entrada sin procesar no ocurre nada, en cambio, si está procesada, habrá que eliminar los productos | |

| **UC-18** | **Dar de alta usuario** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad de producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente dar de alta a otro usuario en el sistema* | |
| **Precondición** | El usuario desea introducir un nuevo usuario en el sistema | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de seguridad/usuario nuevo |
| | 2 | El sistema carga la tabla con el formulario necesario para crear un nuevo usuario |
| | 3 | El empleado rellena el formulario y pulsa el botón de crear usuario |
| | 4 | El sistema verifica toda la información e inserta el usuario en la estructura del almacén |
| **Postcondición** | Se ha insertado un usuario nuevo | |
| **Excepciones** | **Paso** | **Acción** |
| | 4 | Si el empleado introduce algún dato incorrecto o inconsistente el sistema se lo notifica |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-19** | **Dar de baja usuario** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad de producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente dar de baja a otro usuario en el sistema* | |
| **Precondición** | El usuario desea dar de baja un usuario en el sistema | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de seguridad/usuarios |
| | 2 | El sistema carga la tabla con todos los usuarios existentes en el sistema |
| | 3 | El empleado selecciona el usuario y pulsa el botón de eliminar |
| | 4 | El sistema despliega una ventana de confirmación para eliminar el usuario |
| | 5 | El empleado confirma que quiere borrar el usuario |
| | 6 | El sistema modifica la estructura y borra el usuario |
| **Postcondición** | Se ha dado de baja un usuario | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

<table>
<tr><td>UC-20</td><td colspan="2">Asignar rol usuario</td></tr>
<tr><td>Objetivos asociados</td><td colspan="2">• OBJ-03 Trazabilidad del producto</td></tr>
<tr><td>Requisitos asociados</td><td colspan="2">• IRQ-03 Información sobre trazabilidad de producto</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema debe comportarse tal y como se va a describir a continuación cuando un usuario intente asignar o modificar un rol a otro usuario</td></tr>
<tr><td>Precondición</td><td colspan="2">El usuario debe existir antes de poder asignarle un rol</td></tr>
<tr><td rowspan="7">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El empleado accede al apartado de seguridad/roles</td></tr>
<tr><td>2</td><td>El sistema carga la tabla con los diferentes roles existentes en el sistema</td></tr>
<tr><td>3</td><td>El empleado selecciona el rol que desee y pulsa añadir usuario</td></tr>
<tr><td>4</td><td>El sistema despliega una ventana que permite al empleado seleccionar un usuario</td></tr>
<tr><td>5</td><td>El empleado selecciona al usuario deseado y pulsa el botón asignar</td></tr>
<tr><td>6</td><td>El sistema asigna el usuario al rol previamente seleccionado</td></tr>
<tr><td>Postcondición</td><td colspan="2">Se ha insertado un rol al usuario</td></tr>
<tr><td rowspan="2">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>-</td><td></td></tr>
<tr><td rowspan="2">Rendimiento</td><td>Paso</td><td>Cota de tiempo</td></tr>
<tr><td>2</td><td>1 segundos</td></tr>
<tr><td>Frecuencia</td><td colspan="2">Aprox. 2 veces/mes</td></tr>
<tr><td>Estabilidad</td><td colspan="2">Alta</td></tr>
<tr><td>Comentarios</td><td colspan="2">ninguno</td></tr>
</table>

| **UC-21** | **Quitar rol usuario** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad de producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente eliminar un rol a otro usuario* | |
| **Precondición** | El usuario debe tener asignado el rol | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de seguridad/roles |
| | 2 | El sistema carga la tabla con los diferentes roles existentes en el sistema |
| | 3 | El empleado selecciona el rol que desee y pulsa eliminar usuario |
| | 4 | El sistema despliega una ventana que permite al empleado seleccionar un usuario |
| | 5 | El empleado selecciona al usuario deseado y pulsa el botón eliminar |
| | 6 | El sistema desasigna el usuario al rol previamente seleccionado |
| **Postcondición** | Se ha desasignado un rol a un usuario | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

<table>
<tr><td>UC-22</td><td colspan="2">Asignar permisos de página a rol</td></tr>
<tr><td>Objetivos asociados</td><td colspan="2">• OBJ-03 Trazabilidad del producto</td></tr>
<tr><td>Requisitos asociados</td><td colspan="2">• IRQ-03 Información sobre trazabilidad de producto</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema debe comportarse tal y como se va a describir a continuación cuando un usuario intente asignar o modificar un permiso de página a un rol.</td></tr>
<tr><td>Precondición</td><td colspan="2">Antes de asignar permisos de página a un rol, este debe estar previamente creado</td></tr>
<tr><td rowspan="7">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El empleado accede al apartado de seguridad/roles</td></tr>
<tr><td>2</td><td>El sistema carga la tabla con los diferentes roles existentes en el sistema</td></tr>
<tr><td>3</td><td>El empleado selecciona el rol que desee y pulsa añadir página</td></tr>
<tr><td>4</td><td>El sistema despliega una ventana que permite al empleado seleccionar una página de la aplicación</td></tr>
<tr><td>5</td><td>El empleado selecciona la página deseada y pulsa el botón asignar</td></tr>
<tr><td>6</td><td>El sistema asigna permisos sobre una página al rol previamente seleccionado</td></tr>
<tr><td>Postcondición</td><td colspan="2">Se han insertado permisos de página a un rol</td></tr>
<tr><td rowspan="2">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>-</td><td></td></tr>
<tr><td rowspan="2">Rendimiento</td><td>Paso</td><td>Cota de tiempo</td></tr>
<tr><td>2</td><td>1 segundos</td></tr>
<tr><td>Frecuencia</td><td colspan="2">Aprox. 2 veces/mes</td></tr>
<tr><td>Estabilidad</td><td colspan="2">Alta</td></tr>
<tr><td>Comentarios</td><td colspan="2">ninguno</td></tr>
</table>

| **UC-23** | **Quitar permisos de página a rol** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-03 Trazabilidad del producto | |
| **Requisitos asociados** | • IRQ-03 Información sobre trazabilidad de producto | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente desasignar o modificar un permiso de página a un rol.* | |
| **Precondición** | Antes de desasignar permisos de página a un rol, este debe de tener el permiso de página asignado | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de seguridad/roles |
| | 2 | El sistema carga la tabla con los diferentes roles existentes en el sistema |
| | 3 | El empleado selecciona el rol que desee y pulsa eliminar página |
| | 4 | El sistema despliega una ventana que permite al empleado seleccionar una página de la aplicación |
| | 5 | El empleado selecciona la página deseada y pulsa el botón desasignar |
| | 6 | El sistema desasigna permisos sobre una página al rol previamente seleccionado |
| **Postcondición** | Se han eliminado permisos de página a un rol | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 1 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

<table>
<tr><td>UC-24</td><td colspan="2">Eliminar entrada</td></tr>
<tr><td>Objetivos asociados</td><td colspan="2">• OBJ-01 Gestionar las entradas de productos</td></tr>
<tr><td>Requisitos asociados</td><td colspan="2">• IRQ-01 Información sobre entradas</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema debe comportarse tal y como se va a describir a continuación cuando un usuario intente eliminar una entrada en el almacén.</td></tr>
<tr><td>Precondición</td><td colspan="2">El usuario quiere eliminar una entrada del almacén</td></tr>
<tr><td rowspan="7">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El empleado accede al apartado de entradas</td></tr>
<tr><td>2</td><td>El sistema muestra una tabla con la información de las entradas existentes en el sistema</td></tr>
<tr><td>3</td><td>El empleado pulsa el botón de eliminar entrada que se muestra en la página entradas</td></tr>
<tr><td>4</td><td>El sistema muestra una ventana con un botón de confirmación de borrado</td></tr>
<tr><td>5</td><td>El empleado pulsa el botón y confirma el borrado de la entrada</td></tr>
<tr><td>6</td><td>El sistema borra la entrada y realiza las modificaciones necesarias</td></tr>
<tr><td>Postcondición</td><td colspan="2">Se ha eliminado una entrada</td></tr>
<tr><td rowspan="2">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>-</td><td></td></tr>
<tr><td rowspan="2">Rendimiento</td><td>Paso</td><td>Cota de tiempo</td></tr>
<tr><td>6</td><td>2 segundos</td></tr>
<tr><td>Frecuencia</td><td colspan="2">Aprox. 2 veces/mes</td></tr>
<tr><td>Estabilidad</td><td colspan="2">Alta</td></tr>
<tr><td>Comentarios</td><td colspan="2">ninguno</td></tr>
</table>

| **UC-24** | **Eliminar tipos de ubicaciones** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente eliminar un tipo de ubicación en el almacén* | |
| **Precondición** | El usuario quiere eliminar un tipo de ubicación del almacén | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de tipos de ubicaciones |
| | 2 | El sistema muestra una tabla con la información de los tipos de ubicaciones existentes en el almacén |
| | 3 | El empleado pulsa el botón de eliminar tipo de ubicación que se muestra en la página tipos de ubicaciones |
| | 4 | El sistema muestra una ventana con un botón de confirmación de borrado |
| | 5 | El empleado pulsa el botón y confirma el borrado del tipo de ubicación |
| | 6 | El sistema borra el tipo de ubicación y realiza las modificaciones necesarias |
| **Postcondición** | Se ha eliminado un tipo de ubicación | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 6 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

| **UC-26** | **Insertar ubicaciones** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente insertar una ubicación en el almacén*. | |
| **Precondición** | El usuario quiere insertar una ubicación | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de ubicaciones |
| | 2 | El empleado pulsa el botón de insertar ubicación que se muestra en la página ubicaciones |
| | 3 | El sistema muestra una ventana en la que el empleado debe introducir los datos necesarios. |
| | 4 | El empleado rellena los datos y confirma la inserción |
| | 5 | El sistema introduce los datos en el almacén |
| **Postcondición** | Se ha insertado una nueva ubicación | |
| **Excepciones** | **Paso** | **Acción** |
| | 4 | Si la ubicación ya existe en el sistema se le comunica al empleado |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 5 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

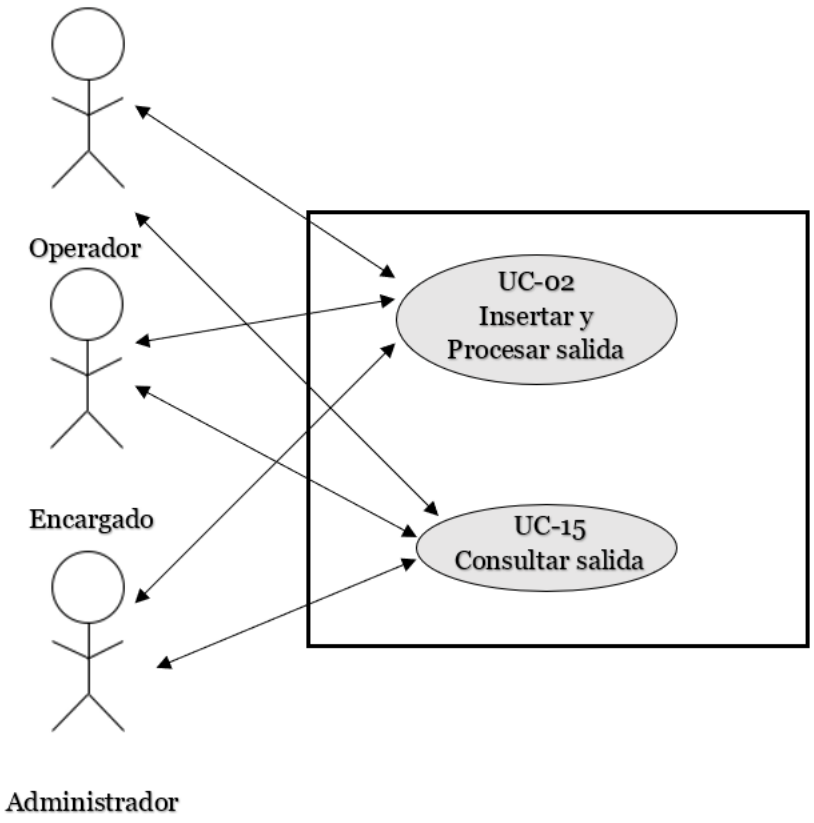
| **UC-27** | **Eliminar ubicaciones** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-04 Gestión del stock | |
| **Requisitos asociados** | • IRQ-04 Información de stock | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente eliminar una ubicación en el almacén* | |
| **Precondición** | El usuario quiere eliminar una ubicación del almacén | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de ubicaciones |
| | 2 | El sistema muestra una tabla con la información de las ubicaciones existentes en el almacén |
| | 3 | El empleado pulsa el botón de eliminar ubicación que se muestra en la página ubicaciones |
| | 4 | El sistema muestra una ventana con un botón de confirmación de borrado |
| | 5 | El empleado pulsa el botón y confirma el borrado de la ubicación |
| | 6 | El sistema borra la ubicación y realiza las modificaciones necesarias |
| **Postcondición** | Se ha eliminado una ubicación | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 6 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

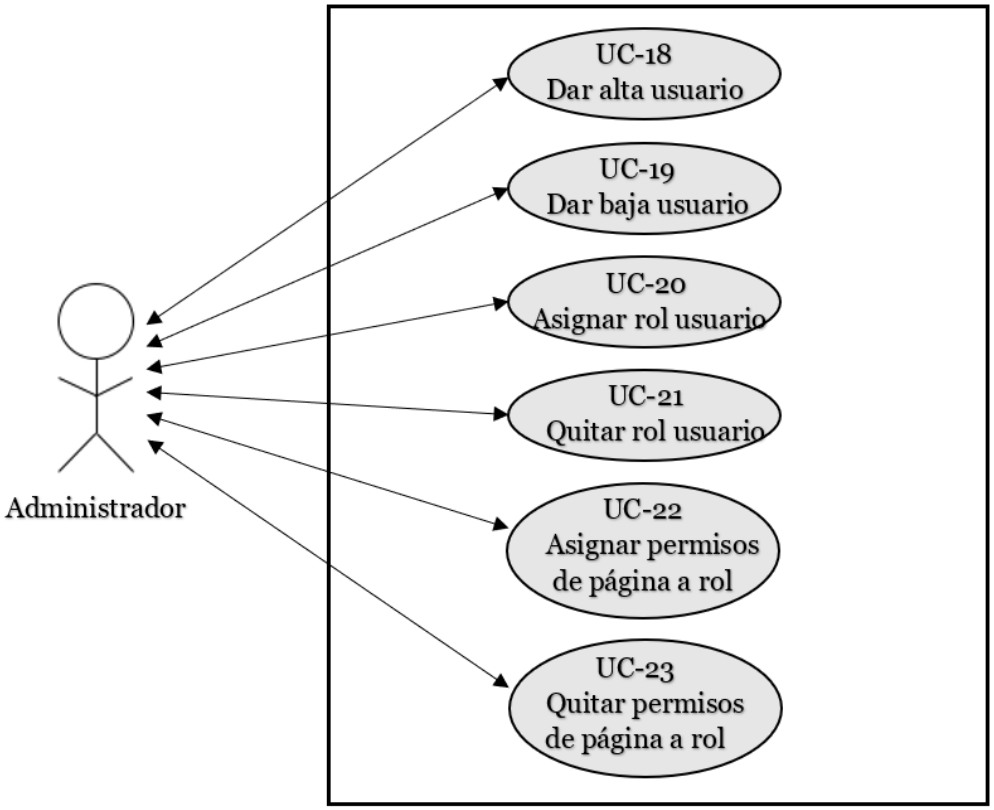
| **UC-28** | **Insertar parte de trabajo** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-05 Gestionar partes de trabajo | |
| **Requisitos asociados** | • IRQ-05 Información de partes de trabajo | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente insertar un parte de trabajo* | |
| **Precondición** | El usuario quiere insertar un parte de trabajo en el sistema | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de partes, nuevo parte |
| | 2 | El sistema muestra un formulario con todos los campos que se tienen que rellenar para crear un parte |
| | 3 | El empleado rellena los campos del formulario y pulsa el botón insertar parte |
| | 4 | El sistema comprueba los datos y si son correctos los inserta en la base de datos |
| **Postcondición** | Se ha insertado un parte de trabajo | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 4 | 1 segundos |
| **Frecuencia** | Aprox. 80 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

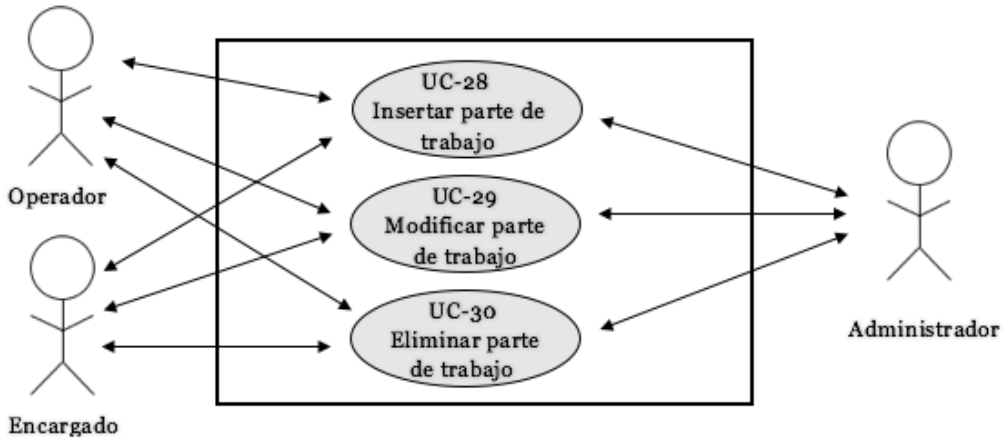
| **UC-29** | **Modificar parte de trabajo** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-05 Gestionar partes de trabajo | |
| **Requisitos asociados** | • IRQ-05 Información de partes de trabajo | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente modificar un parte de trabajo* | |
| **Precondición** | El usuario quiere modificar un parte de trabajo en el sistema | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de partes |
| | 2 | El sistema muestra un resumen de todos los partes que se han registrado en la base de datos |
| | 3 | El empleado pulsa el botón modificar parte en el parte que quiera modificar |
| | 4 | El sistema muestra los campos del parte que son modificables |
| | 5 | El empleado rellena los campos que quiere modificar y pulsa el botón modificar |
| | 6 | El sistema registra los nuevos datos |
| **Postcondición** | Se ha modificado un parte de trabajo | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |

| **UC-30** | **Eliminar parte de trabajo** | |
|---|---|---|
| **Objetivos asociados** | • OBJ-05 Gestionar partes de trabajo | |
| **Requisitos asociados** | • IRQ-05 Información de partes de trabajo | |
| **Descripción** | El sistema debe comportarse tal y como se va a describir a continuación *cuando un usuario intente eliminar un parte de trabajo* | |
| **Precondición** | El usuario quiere modificar un parte de trabajo en el sistema | |
| **Secuencia normal** | **Paso** | **Acción** |
| | 1 | El empleado accede al apartado de partes |
| | 2 | El sistema muestra un resumen de todos los partes que se han registrado en la base de datos |
| | 3 | El empleado pulsa el botón eliminar parte en el parte que quiera modificar |
| | 4 | El sistema muestra una ventana para que el empleado confirme la eliminación |
| | 5 | El empleado acepta la eliminación |
| | 6 | El sistema elimina el parte del sistema |
| **Postcondición** | Se ha eliminado un parte de trabajo | |
| **Excepciones** | **Paso** | **Acción** |
| | - | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 2 | 2 segundos |
| **Frecuencia** | Aprox. 2 veces/mes | |
| **Estabilidad** | Alta | |
| **Comentarios** | ninguno | |

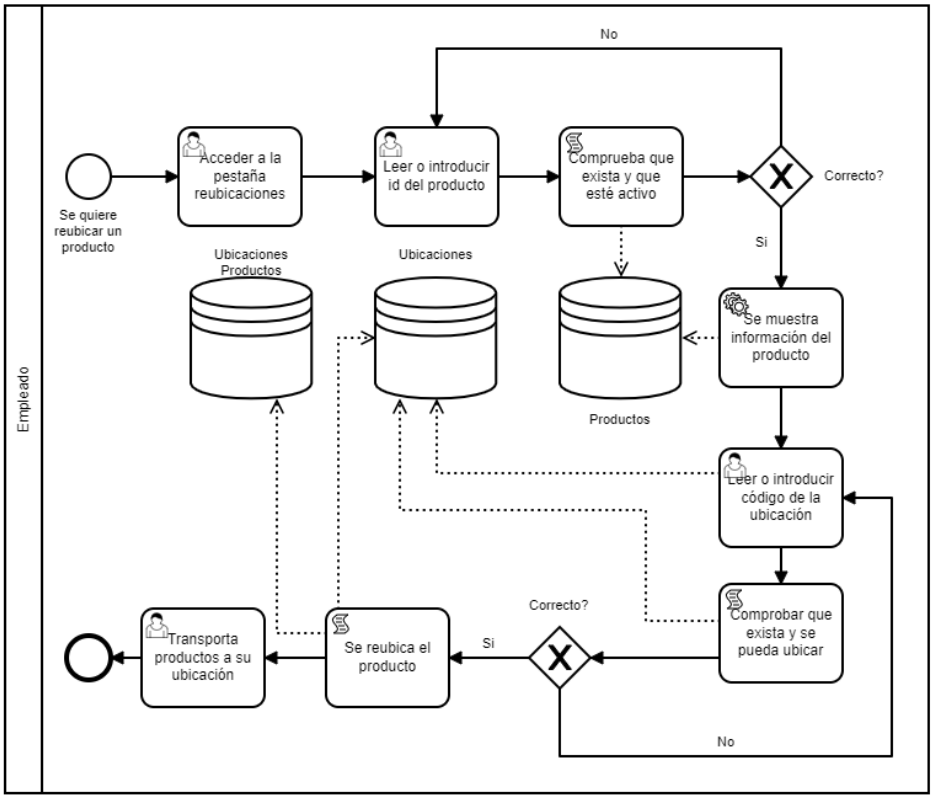
## A.4 Diagramas de casos de uso

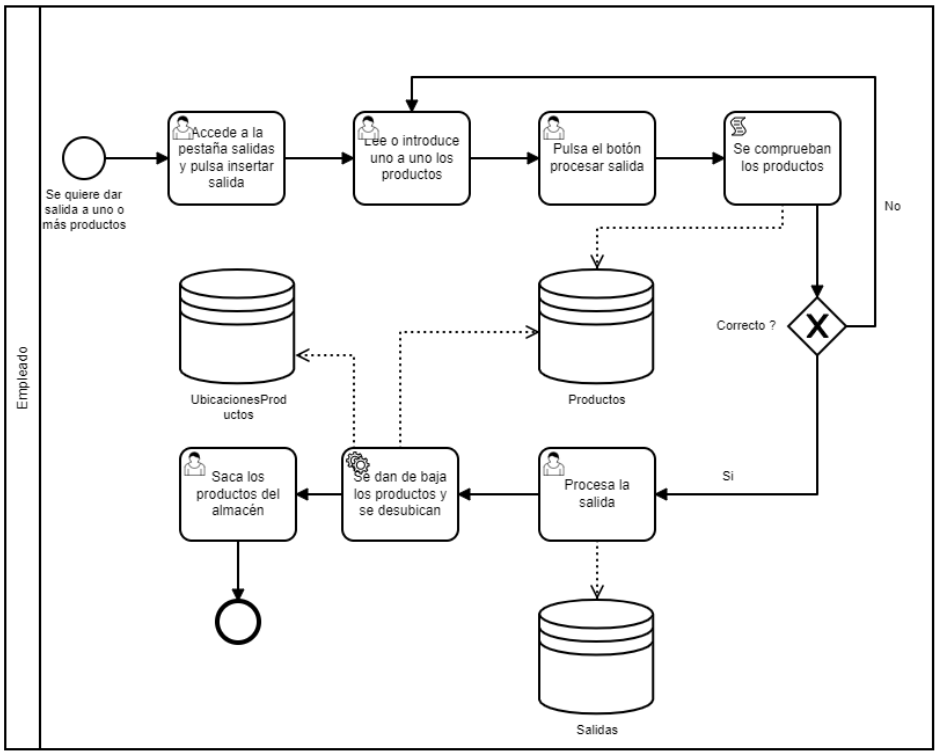

Operador
Encargado
Administrador
UC-02
Insertar y
Procesar salida
UC-15
Consultar salida


Administrador
UC-18
Dar alta usuario
UC-19
Dar baja usuario
UC-20
Asignar rol usuario
UC-21
Quitar rol usuario
UC-22
Asignar permisos
de página a rol
UC-23
Quitar permisos
de página a rol


Operador
Encargado
UC-28
Insertar parte de trabajo
UC-29
Modificar parte de trabajo
UC-30
Eliminar parte de trabajo
Administrador

# B Diagramas de flujo



# C Esquema relacional