Intranet PYME logística

Iván Rubí Lozano

Resumen— Aplicación web de una intranet enfocada para un entorno empresarial logístico. La aplicación permite un registro de usuarios, llevar un control inventariado del material disponible, poder abrir y resolver tickets de incidencias, crear eventos globales para posteriormente visualizarlos en un calendario mensual y crear entradas en un tablón de mensajes. El proyecto ha sido desarrollado de forma modular con tecnologías maduras muy utilizadas. Es una aplicación que permite ayudar a PYMEs logísticas con pocos recursos las cuales necesitan de este tipo de intranets de gestión para poder llegar a ser más eficientes en su trabajo. Al estar desarrollada de manera modular se puede adaptar a todo tipo de clientes logísticos los cuales necesitan módulos específicos.

Paraules clau—Intranet, MVC, aplicación web, logística, jQuery, Javascript, Symfony, PYME.

Resum— Aplicació web d'una intranet enfocada per a un entorn empresarial logístic. L'aplicació permet un registre d'usuaris, portar un control inventariat del material disponible, poder obrir i resoldre tiquets d'incidències, crear esdeveniments globals per a posteriorment visualitzar-los en un calendari mensual i crear entrades en un tauló de missatges. El projecte ha estat desenvolupat de forma modular amb tecnologies madures molt utilitzades. És una aplicació que permet ajudar a PYMEs logístiques amb pocs recursos les quals necessiten d'aquesta mena d'intranets de gestió per a poder arribar a ser més eficients en el seu treball. Com està desenvolupada de manera modular es pot adaptar a tota mena de clients logístics els quals necessiten mòduls específics.

Paraules clau—Intranet, MVC, aplicació web, logística, jQuery, JavaScript, Symfony, PYME.

Abstract—Web application of an intranet focused for logistics business environments. This app allows to register users, to check up on the available inventory, to open and solve queries, to create events that will appear on the monthly calendar and to create new messages on a message board. This project has been carried out in a modular way with intricate and widely used technology. It is an app that intends to help SME with scarce means that need this kind of intranet to enhance their efficiency and productivity. Because the app has been developed in a modular way, it is easily adjusted to different types of customers with different needs.

Index Terms— Intranet, MVC, web app, logistic, jQuery, Javascript, Symfony, SEM.

1 INTRODUCCIÓN

No son pocas las empresas que necesitan de un software de gestión propio. Construir y mantener una intranet a medida puede suponer un gasto innecesariamente alto para una entidad corporativa, especialmente las PYMEs, las cuales tienen que desechar esta idea a pesar de las muchas ventajas que puede llegar a aportar a la productividad de la empresa. El proyecto se basa en proporcionar un servicio personalizado y más ligero que se adapta a las necesidades del cliente. Este software va a estar enfocado a PYMEs logísticas, las cuales suelen disponer de un presupuesto bajo y carecen de cualquier tipo de software interno, más que el de controlar los pedidos. Para conseguir este propósito, el software ha sido desarrollado de forma modular y de manera escalable siguiendo unos principios SOLID. Con esto se puede conseguir no solo un único target, sino ampliar más el posible mercado, desde las PYME que solo puedan permitirse unos módulos básicos hasta entidades grandes que pueden permitirse tener funcionalidades más específicas que les sean realmente útiles.

En las siguientes páginas se explicará con más detalle en qué consiste este tipo de desarrollo y se hablará tanto de los objetivos que se han podido lograr como de los que se han tenido que desestimar.

Se expondrá la metodología de trabajo seguida y la arquitectura que se ha diseñado para este sistema.

Por último, se harán demostraciones de resultados de las diferentes funcionalidades del software basándonos en casos de uso específicos y sus tests.

2 OBJETIVOS

En esta sección se van a detallar los objetivos que se han planteado para el proyecto y la evolución que estos han tenido. Vamos a hablar tanto de objetivos realizados como desestimados, pasando por objetivos planteados a futuro.

2.1 Objetivos Primordiales

A continuación, se muestran los objetivos principales ordenados en formato tabla. Dentro de cada objetivo se explica cómo se va a cumplir.

Tabla 1
Objetivos primordiales y su definición

1. Desarrollo modular
Como he citado anteriormente, el principal objetivo a la hora de desarrollar este software es que sea modular. Esto me va a permitir crear controladores y vistas las cuales no sean dependientes de otras.
1.1 Patrón MVC PHP
Se utiliza el patrón Modelo Vista Controlador para poder realizar la implementación de forma modular. El desarrollo del backend se realiza en PHP con el framework Symfony 4.2
1.2 Vistas con templates
Las vistas son desarrolladas con el motor de templates Twig. Esto me permite poder renderizar todo el contenido de los controladores a las vistas de una manera más directa. Además, se pueden reaprovechar partes del código sin tener que replicarlo, solo hay que llamar a ese trozo de código donde quieras que se muestre.
2. Facilidad de uso
El posible target de este software puede variar muchísimo. Desde gente joven a personas más mayores, con más o menos conocimientos de informática. Es por ello por lo que un objetivo importante es el de desarrollar un software fácil de utilizar con un diseño simple pero eficaz.
2.1 Limitar contenidos
Poco es mucho. El objetivo es conseguir no dar todas las opciones de golpe, solo mostrar lo que en un principio se vaya a necesitar.
2.2 Estándares de diseño
Se seguirán estándares más comunes de diseño web para que sea lo más intuitiva posible.
3. Escalabilidad
Otro objetivo que ha ido apareciendo es el de crear un software con capacidad de poder crecer tanto como sea necesario.
3.1 Controladores CRUD
En el backend se utiliza el diseño CRUD para desarrollar los controladores. Será mucho más sencillo poder ir añadiendo nuevos controladores de esta manera, si en un futuro más gente participa en el proyecto será un proceso de adaptación rápido.
3.2 Principios SOLID
Un software escalable es en el que, aparte de poder añadir más funcionalidades sin que afecten al resto, hace crecer el equipo técnico de desarrolladores disminuyendo el tiempo de estudio del código. Por ello, se seguirán los principios SOLID, unas reglas escritas sobre cómo desarrollar correctamente un código.
4. Flexibilidad
Este software, gracias a seguir un desarrollo modular, podemos adaptarlo a las necesidades del cliente. Retirar módulos innecesarios y añadir otros más oportunos.
4.1 Docker
Para poder aportar esa flexibilidad buscada en el proyecto se van a dockerizar los servicios de este. Con esto, conseguimos poder desplegar el software en cualquier entorno sin tener en cuenta el sistema operativo del cliente.
5. Seguridad
Es importante desarrollar un software seguro y bien testeado, ahorrará muchos problemas por parte del cliente en un futuro. Puede que haga parecer el proceso un poco más lento, pero a la larga se ahorra faena de resolución de incidencias.
5.1 Tests
El software ha sido sometido a una serie de tests con la herramienta PHPUnit. Se han realizado tanto tests de caja blanca como de caja negra, tests unitarios, pathcoverage…
5.2 Consultas parametrizadas
El hacking está a la orden del día. Cierto es que al ser una intranet a la cual solo se podrá acceder desde la propia red del cliente, limita mucho la posibilidad de ser atacada. Aun así, todas las consultas han sido parametrizadas para evitar posibles ataques por este lado.
5.3 Validación de datos
Se realizará una validación de datos que pueda introducir el usuario tanto en la parte del cliente como en la parte del servidor.
5.4 Login y registro seguro
El login se realizará con tokens que el propio Symfony se encarga de generar para aumentar así la seguridad de la sesión del usuario. Para el registro se utilizará un cifrado en las contraseñas, en la bbdd solo se guardará el password cifrado.

2.2 Objetivos Secundarios

A continuación, se muestran los objetivos secundarios que han ido surgiendo a medida que el proyecto avanzaba.

Tabla 2 .Objetivos secundarios y su explicación

Objetivo	Explicación
Filtrar información	En las tablas se usó la función DataTable de la librería jQuery para poder tener tablas dinámicas en las vistas. De esta manera, las tablas tienen paginación, un buscador u ordenación de elementos por columnas.
Docker Compose	En vez de administrar los contenedores con diferentes Dockerfiles e ir haciendo un despliegue de cada uno de ellos, se ha decidido utilizar el fichero Docker compose para poder administrar desde un único punto todos los contenedores.
Webpack encore	Symfony trae webpack encore, es un wrapper que nos facilita el desarrollo de frontend. Con este, podemos trabajar con CSS y JS de una manera más limpia. Podemos programar módulos de JS y luego importarlos donde necesitemos.
Eventos desplazables	Los eventos del calendario se pueden desplazar dinámicamente. Solo es necesario mantener pulsado un evento y arrastrarlo con el ratón donde se desee.
Adquirir material	En el módulo de inventariado de material se ha añadido una funcionalidad extra con la que una persona puede hacer uso de un material y luego devolverlo. En futuras secciones se explica más detalladamente.
Reabrir incidencias	Esta funcionalidad fue una aportación de mi tutora, que sugirió que existiera la posibilidad de reabrir un ticket que había sido previamente cerrado. De manera que, si una incidencia se ha vuelto a repetir, no hay por qué crear una de nuevo; con tan solo reabrir el ticket es suficiente.

3 ESTADO DEL ARTE

Para llegar a construir esta idea de proyecto realicé una investigación de mercado previamente para saber si era viable llevar a cabo el desarrollo. Para ello, indagué sobre empresas que ofrecieran un software ligero de gestión logística. La búsqueda no fue fructífera, puesto que no pude encontrar ninguna. Sin embargo, la falta de resultados me planteaba un escenario positivo: un buen punto de partida desde el que construir mi proyecto. Seguidamente, mencionaré dos empresas que sí pueden llegar a ofrecer un software de gestión.

3.1 Itop

Itop es un software open source que está más enfocado a servicios tecnológicos. Ofrece un apartado de Helpdesk y tiene 4 tipos de modos de configuración en función de cuánto se vaya a pagar, desde el iTop community – que es gratuito – hasta el iTop profesional –en el que viene todo incluido–.

Figura 1. Logo iTop

3.2 Sap

Figura 2. Logo SAP

Esta empresa se dedica a desarrollar todo tipo de intranets para cualquier tipo de empresas. Destaco de ella su gran capacidad para poder adaptar el software a las necesidades más específicas del cliente. No obstante, no es nada barato contratar este tipo de servicio. Además, también incluye ERP y CRM, que no es el objetivo de este proyecto.

La diferencia que marca mi herramienta con respecto a las que hay en el mercado es su viabilidad de implementación en PYMEs con recursos limitados. Esta herramienta aporta una flexibilidad que no aportan ni iTop ni SAP, ya que ambas tienen paquetes cerrados donde se paga por el paquete y este a veces viene con módulos que al usuario final no le interesa. Mi proyecto pretende hacer frente a ese tipo de negocio creando un software más adaptable al cliente, un software que parte de unos módulos básicos y que puede expandirse en función de sus necesidades. El cliente pagará por la necesidad de módulos o funcionalidades que requiera, no por paquetes predefinidos.

4 METODOLOGÍA

Para este proyecto se está siguiendo una metodología **Kanban** con un nivel bajo-medio de rigor. Compagino mi vida universitaria con una vida profesional la cual me ocupa 40h a la semana, es por eso por lo que seguir otro tipo de metodologías más estrictas en cuanto a horarios, como una metodología basada en sprints, no tendría sentido.
Kan significa "visual" y Ban significa "tarjeta". Básicamente consiste en la creación de tareas que serán publicadas en algún tablón. Puede ser un tablón físico o virtual.

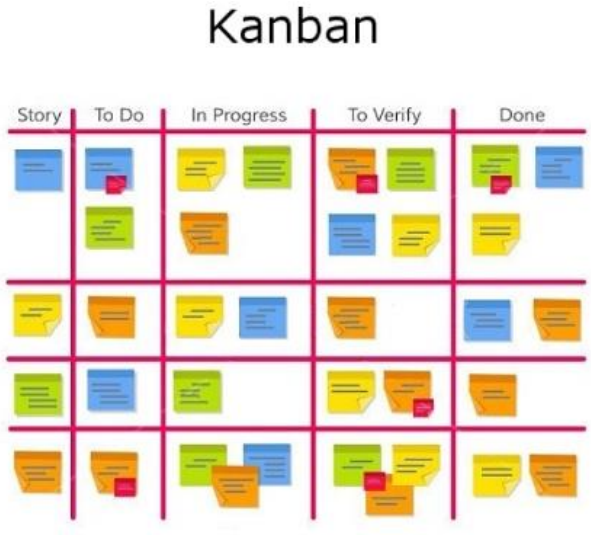

Figura 3. Metodología KanBan

Los procesos por los que pasa una tarea son:

-**Backlog**: Se crea la tarea y queda en el tablón a la espera de ser revisada.
-**Aprobada**: La tarea ha sido aprobada para poder comenzar su desarrollo.
-**En proceso**: La tarea ha sido seleccionada del backlog y comienza su desarrollo.
-**Testing**: La tarea ha terminado su desarrollo y está en fase de testing.
-**Finalizada**: La tarea, una vez desarrollada y testeada, pasa al estado de finalizado.
Al tener un ritmo de vida con poco tiempo disponible, mi metodología se basa en ir añadiendo tareas al backlog y, en cuanto dispongo de tiempo para poder invertir en el proyecto, coger una tarea y hacer que pase por todas sus fases.

5 Planificación

El desarrollo de este proyecto comenzó antes de tiempo, concretamente en enero, ya que fue una idea que tuve para la empresa en la que estaba anteriormente, pero me di cuenta de que podía aplicarla a cualquier empresa del mismo sector.

Al principio, lo que hice fue basar el proyecto en 7 grandes fases y hacer una previsión de cuánto porcentaje de proyecto podía ocupar cada fase.

Figura 4. Timeline fases

Tras un proceso de elicitación pude obtener gran parte de las tareas que serían necesarias realizar para poder tener un producto final. Lo cierto es que a medida que el proyecto ha ido avanzando han surgido nuevas que se han tenido que ir añadiendo.

A continuación, explicaré las fases iniciales, intermedias y finales que conforman las bases sobre las cuales se estructura este proyecto. En el *apéndice 3* se adjunta el diagrama de Gantt de cada fase.

5.1 Fases inicio, elicitación y preparación entorno

Como he comentado anteriormente, este proyecto se comenzó a construir antes de que comenzara el semestre. Las primeras tres fases durarán dos meses: enero y febrero. Normalmente estas fases no suelen requerir de tanto tiempo en proyectos de esta escala, pero como se que tengo cierto margen al comenzar antes, he querido invertir todo este tiempo en construir una buena idea de proyecto controlando cada pequeño matiz.

5.2 Fases diseño, desarrollo y testing

Los meses de marzo y abril serán sin duda los más intensos para el proyecto, puesto que todas las ideas planteadas en las primeras fases deberán implementarse.
Dichas ideas, ahora en forma de tareas, se solapan unas con otras tal y como se puede observar en la *figura 25*. Eso es debido a que la planificación que se seguirá a la hora de desarrollar el software es la de crear módulos y testearlos. No se realizará un nuevo módulo o implementación hasta que el anterior no esté correctamente testeado.

5.3 Fases finales

Estas últimas fases serán menos intensas puesto que el software estará prácticamente terminado. En los meses de mayo y junio mi principal foco será el de terminar los tests que no haya podido realizar y comenzar a desarrollar tanto el penúltimo informe como el final.

6 Tecnologías utilizadas

En esta sección se hablará sobre las distintas tecnologías y herramientas que se han utilizado en el proyecto.

6.1 Docker

Figura 5. Docker logo

Se ha utilizado Docker para generar contenedores de los diferentes servicios que necesita nuestro sistema. Más adelante, en la *sección 7.1*, se explica de manera detallada cómo he implementado este sistema.

6.2 Github

Para el control de versiones se ha utilizado el repositorio Github con su cliente Git para poder trabajar con el propio repositorio de manera local. Además, Github nos facilita una Wiki para poder tener documentado cómo desplegar el proyecto .

Figura 6. Github logo

6.3 Bootstrap

Figura 7. Bootstrap logo

Esta tecnología es un framework para el desarrollo frontend. Nos proporciona una serie de herramientas para personalizar nuestra web con CSS, HTML5 y JavaScript.

6.4 jQuery

Esta herramienta es una librería de JavaScript que me permite interactuar con el DOM con la técnica Ajax. En este proyecto se ha utilizado para poder trabajar con las tablas que muestran datos. Me ha permitido añadir buscadores, filtros y paginación.

Figura 8. jQuery logo

6.5 Symfony PHP

Figura 9. Symfony logo

Para desarrollar el backend, que es la piedra angular de este proyecto, se ha utilizado el framework Symfony en su versión 4.0. Este framework nos proporciona una estructura de carpetas y una serie de herramientas que nos facilita el desarrollo MVC.

6.6 Doctrine ORM

Esta herramienta que nos proporciona el framework Symfony nos permite poder trabajar con nuestra bbdd desde el código, mapeándola y orientándola a objetos. Más adelante en la *sección 7.2* detallo cómo funciona.

Figura 10. ORM

6.7 Twig

Figura 11. TWIG

Esta herramienta es un motor de templates que nos permite escribir código reutilizable en las vistas. Desde un controlador podemos renderizar información a estas vistas y desde ahí tratarlo.

7. Análisis de Requisitos

En este apartado se mostrarán los requisitos tanto funcionales como no funcionales que se especificaron en la fase de elicitación.

7.1 Requisitos no funcionales

En este apartado listaré los no funcionales. Entendemos por no funcionales los requisitos que añaden restricciones en el diseño o la implementación del sistema. Esta arquitectura de sistema, por lo tanto, ha tenido que cumplir con cada uno de estos requisitos.

Tabla 3. Requisitos no funcionales

REQUISITOS	DESCRIPCIÓN DEL REQUISITO
RNF1 - BBDD	La base de datos será construida en MySQL con la capa de abstracción ORM.
RNF2 - Backend	El backend será desarrollado en PHP 7.2.
RNF3 - Frontend	El frontend será desarrollado con HTML, CSS, JS y Twig como motor de templates.
RNF4 - Framework	El sistema será desarrollado con el framework Symfony.
RNF5 - Seguridad	Los passwords serán encriptados con el algoritmo bcrypt y el login se realizará con tokens.
RNF6 - Parametrización	El sistema debe tener todas las consultas parametrizadas para evitar

	cualquier tipo de inyección maliciosa.
RNF7 – Servidor Web	El sistema debe contar con un servidor web Apache para poder acceder al servicio en todo momento.

7.2 Requisitos Funcionales

En este apartado listaré los requisitos funcionales. Entendemos por funcionales las funciones del sistema que deben estar definidas en el software. Estas funciones están descritas como un conjunto de entradas, comportamientos y salidas.

Tabla 4. Requisitos funcionales

REQUISITOS	DESCRIPCIÓN DEL REQUISITO
RF1 - Usuario	El sistema debe ser capaz de registrar un usuario en bbdd con identificador único. Dicho usuario puede realizar un login en el software. El sistema también debe ser capaz de eliminar un usuario creado.
RF2 – Roles	El sistema debe ser capaz de manejar el concepto de roles de usuario. En función del rol asignado a un usuario este tendrá unos permisos u otros en el software.
RF3 – Añadir/eliminar materiales	El sistema debe ser capaz de proporcionar al usuario con rol administrador la capacidad de añadir nuevos materiales de empresa en el módulo "material de empresa". El sistema también debe ser capaz de eliminar materiales creados.
RF4 – Visualizar módulos	El sistema debe proporcionar al usuario la capacidad de poder interactuar con los diferentes módulos implantados.
RF5 – Añadir/eliminar clientes y proveedores	El sistema debe ser capaz de proporcionar al usuario con cualquier rol la capacidad de añadir nuevos clientes y proveedores en el "módulo clientes y proveedores". El sistema también debe ser capaz de eliminar clientes y proveedores creados.
RF6 – Listar incidencias usuario	El sistema debe ser capaz de proporcionar al usuario con cualquier rol la capacidad de listar las incidencias que este haya generado.
RF7 – Listar/resolver incidencias general	El sistema debe ser capaz de proporcionar al usuario con rol administrador la capacidad de listar todas las incidencias generadas por los usuarios. Asimismo, debe ser capaz de proporcionar la posibilidad de resolver dichas incidencias.
RF8 – Añadir/eliminar locales y departamentos	El sistema debe ser capaz de proporcionar al usuario con cualquier rol la capacidad de añadir nuevos locales y departamentos en el "módulo locales y departamentos". Asimismo, debe ser capaz de eliminar locales o departamentos creados.
RF9 – Tablón de mensajes	El sistema debe ser capaz de proporcionar al usuario con cualquier rol la capacidad de añadir un mensaje en el tablón de mensajes, además de poder visualizar los mensajes insertados por otros usuarios.
RF10 - Calendario	El sistema debe ser capaz de proporcionar al usuario con cualquier rol un calendario en el cual poder visualizar los eventos que tendrá la empresa.
RF11 – Eventos de la empresa	El sistema debe ser capaz de proporcionar al usuario con rol administrador la capacidad de añadir eventos que sean volcados en el calendario general. El sistema también debe ser capaz de eliminar dichos eventos.

8 Arquitectura del sistema

En esta sección se va a hablar de cómo está diseñada la arquitectura del software, tanto a nivel de servicios como de código.

8.1 Servicios con Docker

Uno de los objetivos que me planteé fue que el software fuera flexible y escalable. Consideré que la mejor manera de conseguir dicho propósito era la de dockerizar mis servicios. El proceso consiste en crear contenedores independientes los cuales no se ven afectados por el sistema operativo en el que se ejecuten. Esto soluciona muchos problemas de dependencias.

Este sistema está compuesto por 4 contenedores.

-Mysql: Ofrece el servicio de bases de datos.
-Symfony: Es donde está instalado PHP con el frawework.
-Apache: Ofrece el servicio de servidor web HTTP.
-Phpmyadmin: Herramienta para visualizar bbdd.

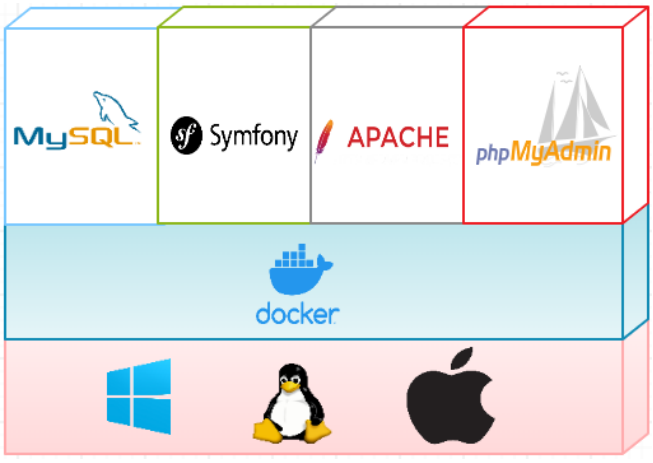

Figura 12. Arquitectura Docker

Gracias a Docker Engine podemos trabajar con estos contenedores a través de un fichero de texto llamado docker-compose. En este fichero definiremos los contenedores y los puertos que tendrán habilitados. Cada vez que queramos arrancar el sistema, con tan solo ejecutar un comando dentro del directorio del proyecto podremos levantar todos los servicios. Por último, como se observa en la *figura 12*, estos contenedores son independientes del sistema operativo, se pueden utilizar tanto en Windows como en Mac o Linux.

8.2 Diseño base de datos MySQL

Para este proyecto se ha decidido utilizar una base de datos relacional construida en MySQL. Esta decisión está fundamentada en la experiencia previa con este tipo de bases de datos.

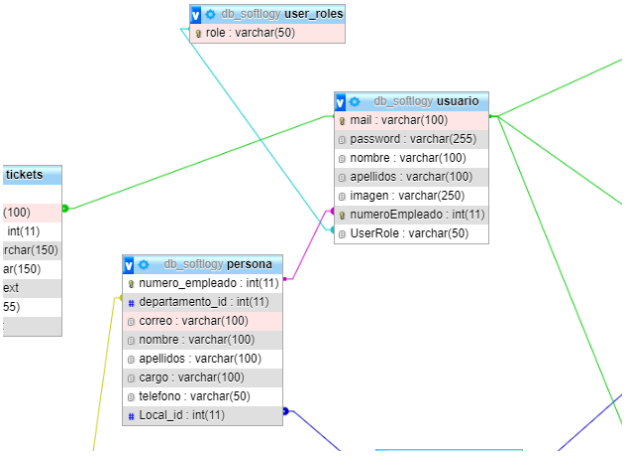

Figura 13. Diseño base de datos MySQL

En el proyecto se han tenido que investigar otras herramientas, así que pensé en la posibilidad de usar una BBDD no relacional como MongoDB, ya que esta ofrece mucha escalabilidad. Sin embargo, con la estimación inicial que realicé en cuanto a necesidad de recursos para investigar las tecnologías que ya estaba utilizando en el proyecto, decidí que era demasiada carga laboral el añadir una más como sería MongoDB.

Las tablas con mayor relación y peso en el software son las de user_roles, usuario y persona que se ven en la *figura 14*. Estas tablas están relacionadas y no se puede crear un usuario sin asignarle una persona y un rol. El resto de las tablas tienen alguna relación con usuario, puesto que toda acción debe quedar registrada. Cada módulo del software tiene su propia tabla en la bbdd. Esto me permite poder añadir de manera sencilla nuevos módulos sin que afecten a los creados anteriormente.

8.3 Arquitectura software MVC

Como he comentado anteriormente, el desarrollo de este software se ha centrado sobre todo en el backend. Se ha trabajado con PHP y el framework Symfony en su versión 4.0. Para poder conseguir el objetivo de un desarrollo modular hay varias arquitecturas que podrían funcionar. En mi caso, me decanté por la arquitectura MVC (Modelo-Vista-Controlador). Esta arquitectura se basa en separar lógica, middleware e información de manera que no esté todo junto en un mismo lugar.

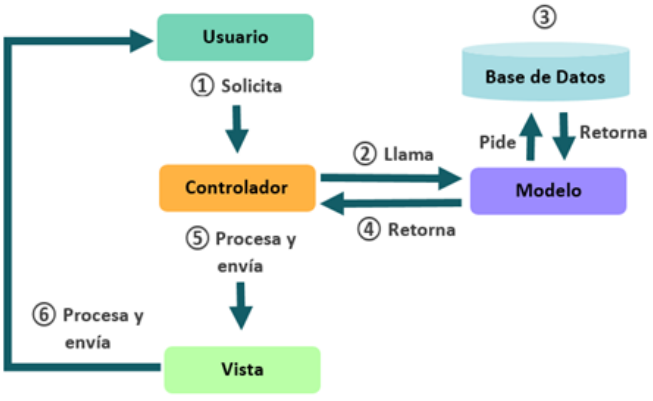

Figura 14. Arquitectura MVC

Como podemos ver en la *Figura 14*, el software se puede dividir en controladores, vistas y modelos. Los controladores tienen toda la lógica del programa, los modelos son los middlewares que actúan entre bbdd y controlador. Por último, la vista tiene toda la información, la cual procesa y envía al cliente, que en este caso sería el usuario.

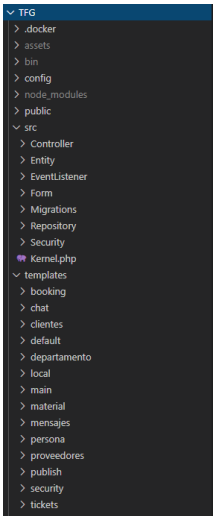

Figura 15. Estructura directorios

Esta es la estructura de directorios del proyecto.
Hay muchos directorios extra de dependencias, Docker, plugins de Symfony…etc.
Lo importante es fijarnos en el directorio *src* y el directorio *templates*. Dentro del directorio *src* podemos ver los controladores en el directorio *controller* y en *entity* podemos encontrar los modelos. Por último, el directorio *templates* tiene las vistas, cuya función es crear un directorio por cada vista. Esta organización me la facilita el framework symfony.

8.4 BBDD con Doctrine ORM

El modelo tradicional del patrón MVC se basa en realizar las consultas a la base de datos que le llegan de peticiones desde el controlador. Para este proyecto se ha decidido añadir una capa de abstracción a la base de datos MySQL con la ayuda de la herramienta Doctrine ORM (Object Relational Mapping) que nos proporciona el framework Symfony. Esta herramienta es parecida a un Mongoose de JavaScript en caso de trabajar con MongoDB.
Lo que nos permite ORM es convertir nuestra BBDD relacional en una base de datos orientada a objetos. Con esto, lo que conseguimos es poder trabajar con la base de datos desde el código pudiendo instanciar sus clases, que serían las tablas, y hacer uso de sus métodos para trabajar con los atributos de las tablas, ya sea insertando, eliminando o buscando registros.

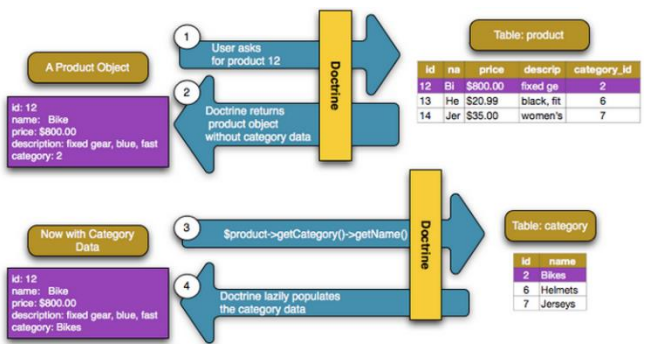

Figura 16. Ejemplo Doctrine ORM

En la *figura 16* se tienen en cuenta dos casos de uso para darle un concepto más práctico a la hora de ver la utilidad de esta herramienta.
Un usuario hace una petición de un producto específico y doctrine devuelve los resultados encapsulándolos en un objeto con 5 atributos. Se crea un segundo ejemplo mostrando el código donde $product es la tabla producto la cual es una instancia de la clase y se puede acceder a sus métodos que serían getCategory() para acceder a los datos del atributo categoría de la tabla productos y getName() para obtener el nombre de esta categoría.

Gracias a esta herramienta se consiguen objetivos como seguridad en el software, ya que no hay código SQL al cual le puedan hacer inyección. También se ahorra duplicación de código, ya que se puede reutilizar al llamar a los métodos de un objeto desde diferentes partes del código, lo cual aporta escalabilidad y flexibilidad.

9. Resultados

En esta sección mostraré los resultados del desarrollo. La dividiré en subsecciones donde por cada una mostraré una funcionalidad desarrollada y una breve explicación de qué aporta al software. Mostraré las funcionalidades con más impacto o las que más carga laboral han tenido para mí.

9.1 Creación de usuario por roles

Para crear un usuario de la intranet se tienen que cumplir dos condiciones. La primera es que previamente se haya creado una entidad persona y la otra es que se le proporcione un rol. Para este proyecto he definido dos roles que son el de role_user y role_admin.

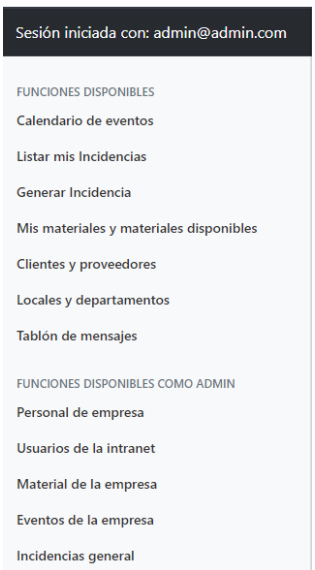

Figura 17. Módulos Admin

Un usuario con privilegios administrador, a diferencia de uno con privilegios normales, puede resolver o cerrar incidencias, crear usuarios, añadir stock y añadir eventos en el calendario.

En el lateral izquierdo de la aplicación se mostrará una lista con las acciones disponibles en función de qué rol se tenga. En la *figura 17* se puede ver que, al tratarse de un usuario administrador, todas están disponibles.

9.2 Resolución tickets

Esta funcionalidad está pensada para una empresa logística la cual tiene un pequeño departamento de resolución de incidencias internas. Un usuario administrador puede entrar en el apartado de "incidencias general" y visualizar qué incidencias hay actualmente abiertas para posteriormente resolverlas. Un usuario sin rol administrador puede generar una incidencia y listar sus incidencias creadas.

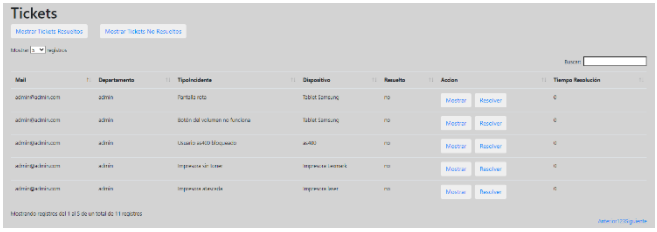

Figura 18. Listado tickets

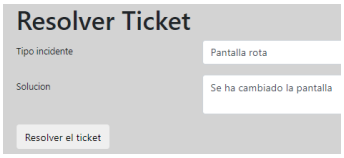

Figura 19. Resolución ticket

En caso de querer resolver una incidencia, el usuario administrador se dirigirá a "resolver", escribirá la solución y así se registrará en el sistema el ticket cerrado.

9.3 Calendario global

Este módulo es accesible por cualquier tipo de usuario. Lo que se pretende con esta funcionalidad es poder organizar los eventos de la empresa de una manera rápida y óptima. Hay empresas logísticas que no tienen implementado un servicio de mensajería interno y el comunicar ciertos eventos o reuniones puede hacerse tedioso. Con este módulo se soluciona ese problema.

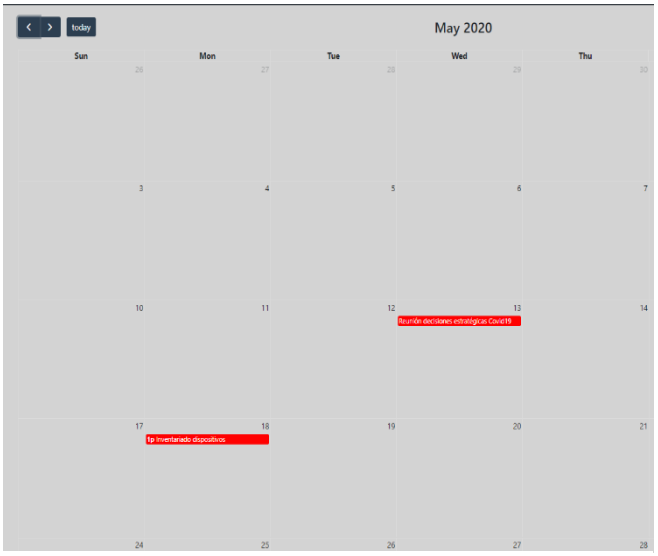

Figura 20. Calendario Global

9.4 Registro material

El objetivo de este módulo es facilitarle a la empresa el poder gestionar el inventario de una manera eficiente. Tiene la opción de clasificar por local por si la empresa tiene más de uno. Este módulo se divide en dos partes:

-Material disponible: Es el material disponible que tiene el local para poder utilizar.
-Mis materiales: Para poder hacer uso de un material, primero hay que entrar en el software y hacer *click* en el botón de agregar. Una vez hecho eso el material se pasará a la lista de mis materiales lo cual significará que el usuario está haciendo uso de ese material y no está disponible para otro usuario.

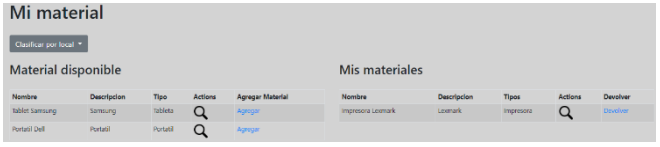

Figura 22. Listado material

En la *figura 22* podemos ver cómo la empresa tiene 3 materiales disponibles para poder utilizar y un cuarto material que está utilizando el usuario actualmente con sesión iniciada. Eso significa que ese usuario estará haciendo uso de ese material y así quedará registrado en el sistema.

10. TEST CON PHPUNIT

Se han realizado tests de caja negra y caja blanca. Symfony proporciona un framework para poder realizar tests que se llama PHPUnit. Este, utiliza assertions para verificar que el comportamiento del código es el esperado. Los tests se han almacenado en el directorio "tests" y se han separado los realizados a controladores y los realizados a los modelos.

10.1 Test caja blanca Statement Coverage

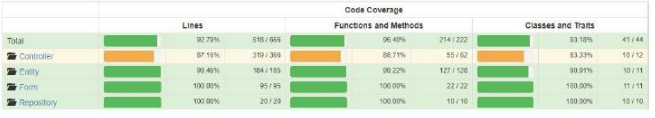

Figura 23. Statement Coverage

Con este test busco verificar que todas las líneas del código importantes han sido testeadas. Gracias a los tests unitarios en el code coverage de la *figura 21* se puede observar cómo las líneas en verde son las testeadas y las rojas las que no. Hay métodos que no pueden ser testeados. Lo comprobamos en los resultados que se ven en la *figura 23*, donde se muestra el porcentaje total de líneas de código recorridas, no es un 100%.

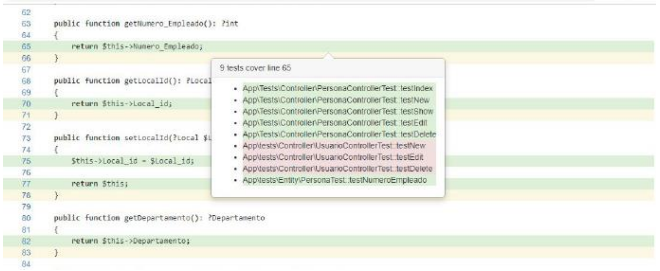

Figura 21. PersonaController tests

10.2 Test caja negra

Lo que intento conseguir con estos tests es buscar que el software cometa un error en algún lado. Es por eso que un 100% de acierto en tests no siempre puede ser bueno, eso tal vez significa que los tests no se han planteado

correctamente.
Para plantearme los tests lo que hice fue ir controlador por controlador y crear una lista escrita de acciones que sí o sí ese módulo debía realizar correctamente. Una vez tenía la lista, generé los valores frontera que podían llegar a hacer que el módulo fallara y lo último fue programar el test que heredaba directamente de la clase que se iba a testear.
En la *figura 24* se muestra un ejemplo de valores frontera interior y exterior donde se comprueba que el nombre de un departamento no exceda los límites establecidos.

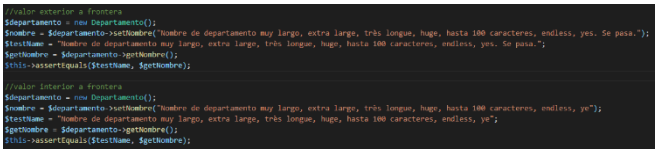

Figura 24. Valores frontera departamento test

11. Conclusiones

Se han cumplido con los objetivos principales del proyecto. Se ha creado un sistema dockerizado con Symfony como backend para la gestión interna de una empresa logística.
La trayectoria del desarrollo ha sido positiva y se han asimilado muchos conceptos nuevos.
Ahora, con un poco más de perspectiva, considero que se podría haber hecho el diseño diferente. Partir de un template de Bootstrap y desarrollar el backend a partir de este para obtener un mejor diseño.
Como líneas futuras marco objetivos que no he podido conseguir durante el desarrollo de este proyecto. Por ejemplo, migrar el software a AWS y utilizar las herramientas de balanceo que este nos ofrece. Comencé a hacer pruebas pero mis necesidades no entraban en el plan gratuito. Otros objetivos a futuro pueden ser crear un chat entre usuarios o implantar librería de diseño de frontend como ReactJS. Estos objetivos se han desestimado por falta de recursos para poderlos llevar a cabo.

12. Agradecimientos

En primer lugar, me gustaría agradecer a mis padres el soporte que me han dado durante estos años de formación académica, por todos los recursos que me han ofrecido sin pedir nada a cambio y por tener la paciencia que han tenido en los momentos más difíciles. Sin su ayuda no podría haber conseguido esta meta tan ambiciosa que es la de graduarse como ingeniero informático. Papa, mama, gracias por haberme dado esta vida, gracias por haberme dado esta oportunidad y gracias por darme un futuro profesional. Espero poder seguir haciendo sentiros orgullosos.

También quiero agradecer el apoyo incondicional de mi pareja, Laura. Compaginar una vida profesional y unos estudios de este grado de dificultad con pareja es algo que puede llegar a ser muy complicado. Laura ha estado a mi lado en todo este duro camino, en las largas noches de biblioteca y largos días de sueño trabajando. Laura me ha ofrecido un apoyo emocional constante, armado de paciencia y cariño. Laura, estas palabras las dirijo directamente a ti, no puedo expresar en un texto tan corto cómo de agradecido estoy por haberme acompañado por este camino lleno de baches, solamente puedo resumirlo en muchas gracias y ahora me toca a mí cuidarte a ti.

No querría pasar por alto mi agradecimiento a Ádrian Viera, compañero de grado superior, de la carrera y compañero de piso. Gracias a él, a tardes y noches de estudio, de debates, de contradicciones, de búsqueda de soluciones… puedo haber podido aprobar asignaturas que tal vez sin su ayuda y su visión no habría podido hacer. Gracias por los consejos amigo, espero haberte aportado como mínimo la misma ayuda que me has aportado tu a mí.

Por último, quiero agradecer a mi tutora, Yolanda Benítez, por su cercanía y predisposición a realizar tantas reuniones como fueran necesarias para poder presentar el proyecto en buenas condiciones. Gracias Yolanda.

13. Bibliografia

[1]https://es.wikipedia.org/wiki/SOLID
[2]http://personales.upv.es/fjabad/pfc/comoEscribir.pdf
[3]https://www.e-intelligent.es/es/blog/ventajas-de-una-intranet-corporativa-para-una-empresa-o-pyme
[4]https://www.itophub.io/
[5]https://www.sap.com/spain/index.html
[6]https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso
[7]https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a
[8]https://aws.amazon.com/es/getting-started/projects/break-monolith-app-microservices-ecs-docker-ec2/
[9] https://es.wikipedia.org/wiki/Diagrama_de_Gantt
[10]https://symfony.com/doc/current/mercure.html
[11] https://swagger.io/solutions/api-documentation/
[12]https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador
[13]https://es.wikipedia.org/wiki/Mapeo_objeto-relacional
[14]https://getbootstrap.com/docs/4.5/getting-started/introduction/
[15]https://twig.symfony.com/
[16]https://symfony.com/doc/current/frontend.html
[17]https://datatables.net/manual/installation
[18]https://symfony.com/doc/current/doctrine.html
[19]https://phpunit.de/getting-started/phpunit-9.html
[20]https://phpunit.readthedocs.io/es/latest/writing-tests-for-phpunit.html
[21]https://jesuslc.com/2013/07/31/phpunit-y-buenas-practicas/
[22]https://xdebug.org/
[23]https://www.elliotjreed.com/post/php/2019-04-03_PHPUnit_Code_Coverage_in_Symfony

APÉNDICE

A1. Índice figuras

A2. Índice tablas

A3. Fases Iniciales

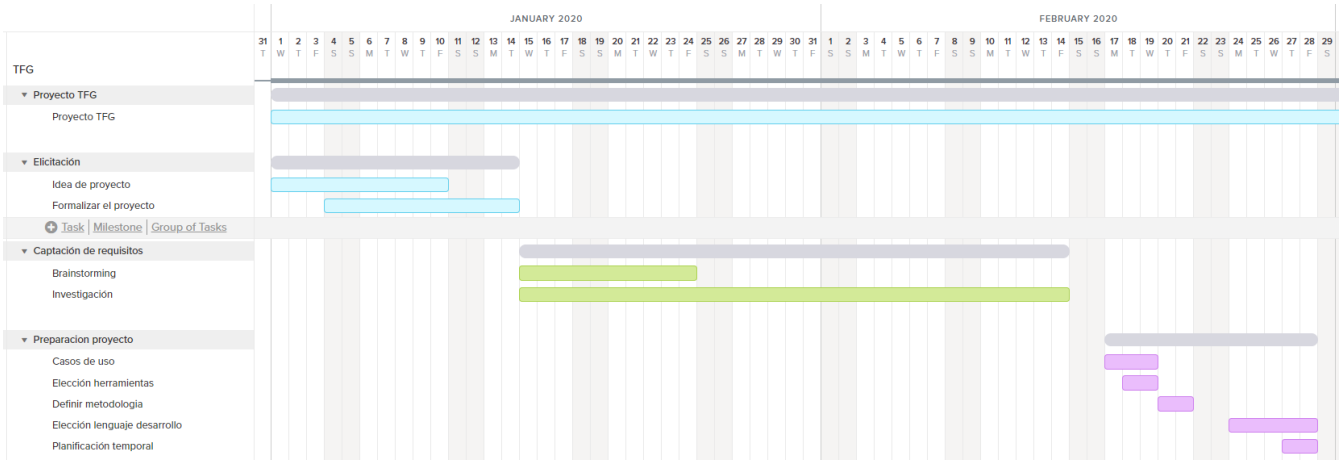

Figura 25. Fases iniciales

A4. Fases Intermedias

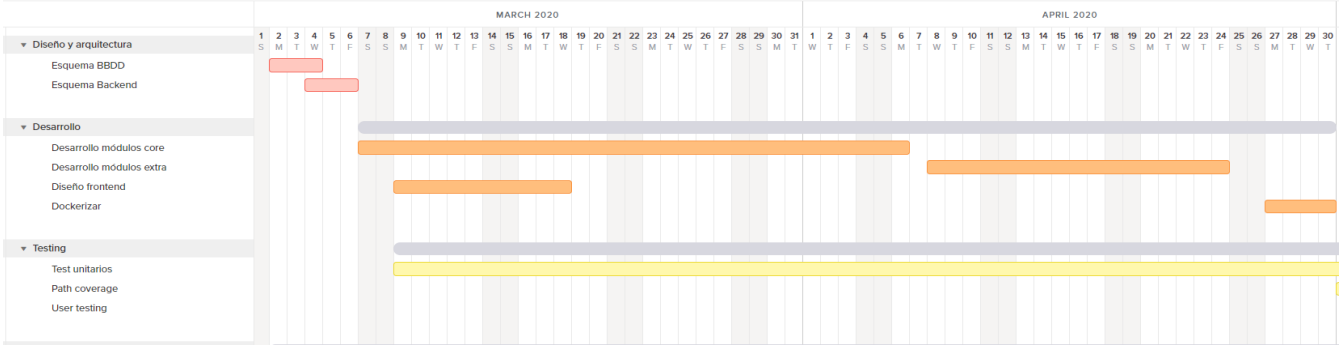

Figura 26. Fases Intermedias

A5. Fases Finales

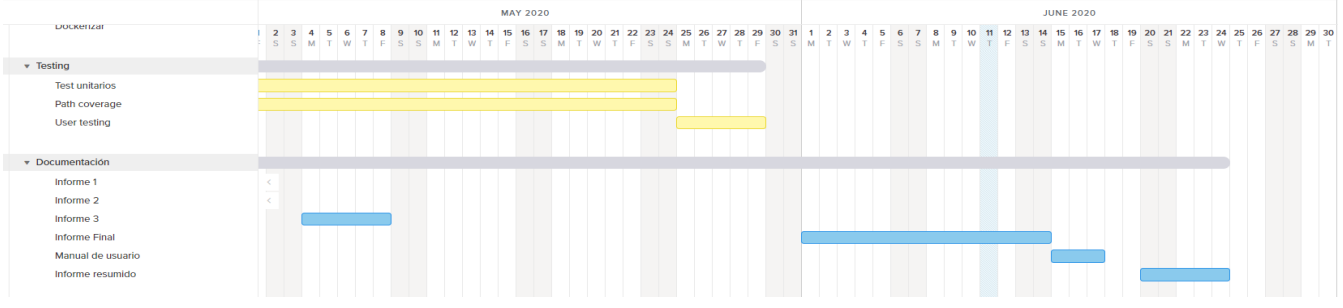

Figura 27. Fases Finales

A6. Diagrama de casos de uso

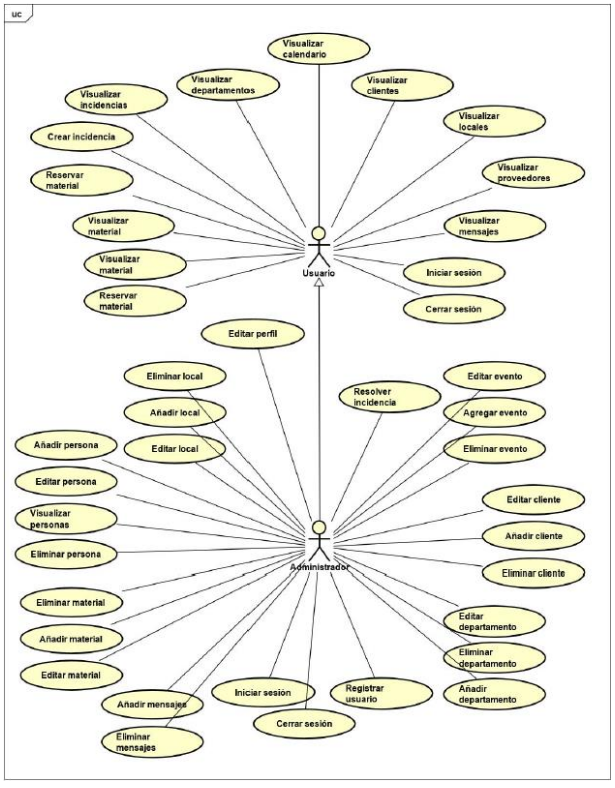

Figura 28. Diagrama de casos de uso

A7. Borradores iniciales

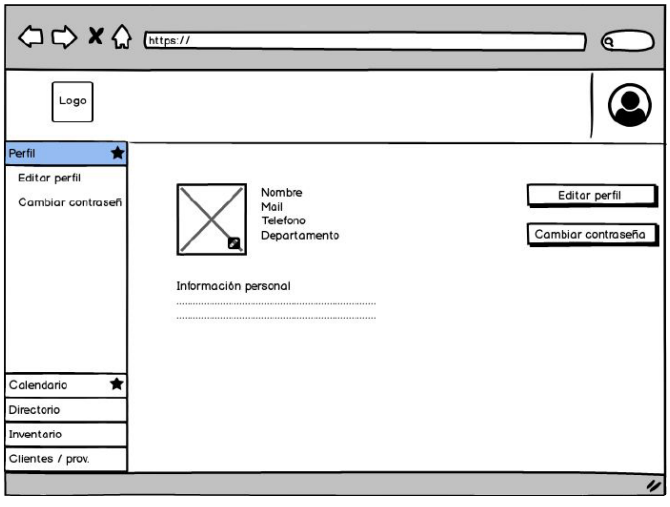

Figura 29. Borrador editar perfil usuario

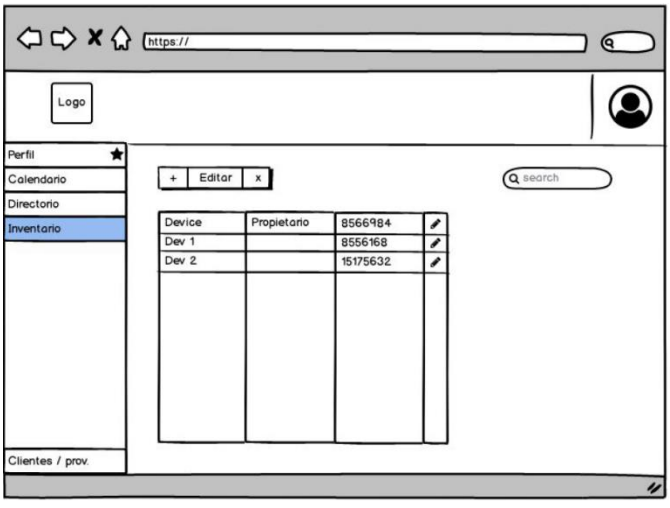

Figura 30. Borrador inventario material

A8. Ejemplo listado comprobaciones test unitario

- Se ha comprobado que el intento de acceso de un usuario no autorizado sea redirigido a la vista Login.
- Se ha comprobado que el intento de acceso de un usuario autorizado se realice correctamente.
- Se ha comprobado que al completar correctamente el formulario para crear un cliente este se crea en la base de datos.
- Se ha comprobado que al completar incorrectamente el formulario para crear un cliente este no se crea en la base de datos.
- Se ha comprobado que al buscar un cliente que existe en la base de datos se obtiene el objeto cliente como resultado.
- Se ha comprobado que al buscar un cliente que no existe en la base de datos se obtiene NULL como resultado.
- Se ha comprobado que al intentar editar los campos de un cliente existente con datos correctos la modificación se realiza correctamente en la base de datos.
- Se ha comprobado que al intentar editar los campos de un cliente existente con datos incorrectos la modificación no se realiza en la base de datos.
- Se ha comprobado que al intentar editar los campos de un cliente que no existe en la base de datos no ocurra nada en la base de datos (caso imposible).
- Se ha comprobado que al intentar borrar un cliente que existe se elimine correctamente de la base de datos.
- Se ha comprobado que al intentar borrar un cliente que no existe en la base de datos no ocurra nada en la base de datos (caso imposible).

A9. Diseño completo BBDD

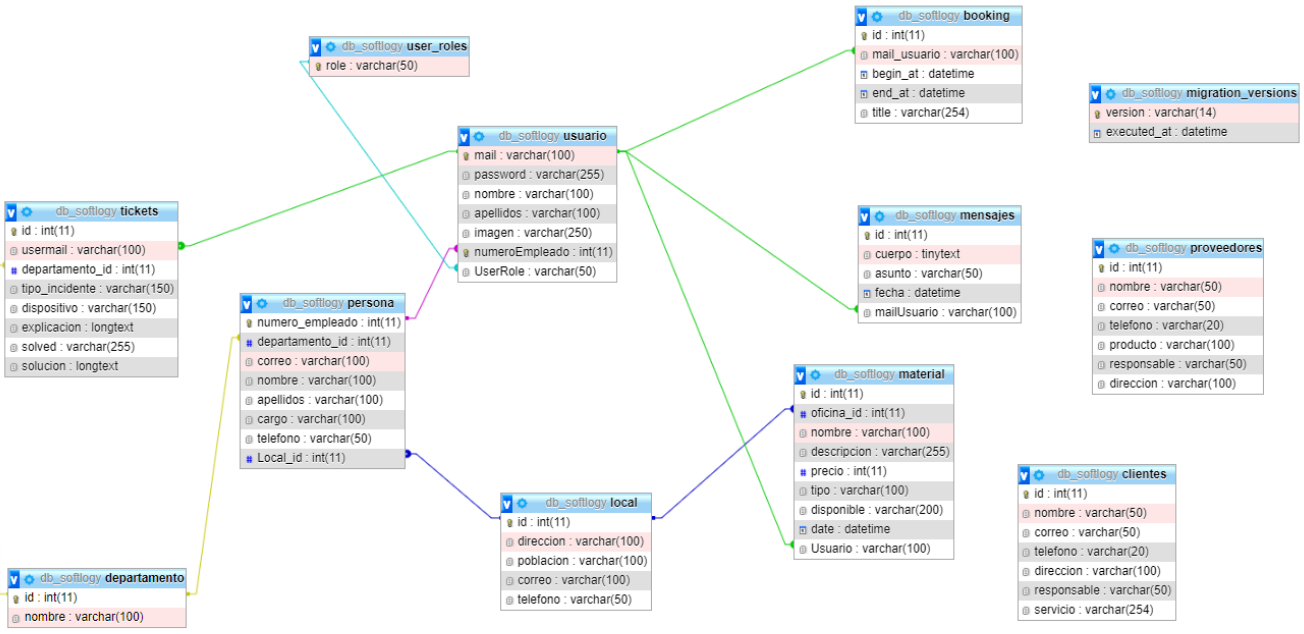

Figura 31. Diseño completo base de datos MySQL