E.T.S. de Ingeniería Industrial, Informática y de Telecomunicación

Sistema integral de gestión de productos, con el proceso de producción, para una empresa de materiales metálicos.

Grado en Ingeniería Informática

Trabajo Fin de Grado

Alumno: Santiago Iribarren Bustince
Director: José Javier Astrain Escola
Pamplona, 11/06/2021

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

En este Trabajo de Fin de Grado se ha realizado un proyecto de implantación de un sistema de gestión de la producción para un cliente concreto. El sistema estará compuesto por varias WEB's y servidor. Consiste principalmente en el desarrollo de un sistema completo de gestión de la producción: desde el consumo de materias primas, hasta la salida de producto final y su expedición. También se desarrollará todo lo referente a la gestión de usuarios y a la gestión de zonas, que más adelante se explicará detalladamente. Por último, se ha replicado una web de checklist digitales

El proyecto se dividirá en cuatro módulos: la página principal Suite, Identity, Blueprint, Checklist y Production. Esta última se divide en tres fases: las entradas de materia prima, el proceso de producción, y las expediciones de los envíos con los paquetes. De cada una de las fases se desarrollará tanto el interfaz web para escritorio como para Tablet, utilizando principalmente el framework de código abierto, denominado como Vue, de JavaScript.

En definitiva, se va a describir el proceso de elaboración de un producto destinado a un cliente dedicado a la fabricación de perfiles metálicos, mayoritariamente de acero.

Palabras Clave

Listado de palabras clave:

- WEB
- JavaScript
- Vue (componentes)
- SASS
- Web Service

Índice

Introducción

iAR (Industrial Augmented Reality)

Industrial Augmented Reality o más conocido como *iAR*, es una empresa fundada en el año 2014 por Jon Navarlaz, director general y cofundador, Ana Monreal, cofundadora, y Miguel Ángel Llorente, cofundador. Se dio a conocer con la instalación de una planta piloto en una empresa de perfiles metálicos. De ahí en adelante, la empresa ha ido evolucionando, y ha sido reconocida con varios premios, como son: el segundo premio en un concurso de ERDF, la compañía eléctrica más importante en Francia; reconocimiento de ANECORM (Asociación Nacional de Enfermería Coordinadora de Recursos Materiales) por un proyecto realizado conjuntamente con el Complejo Hospitalario de Navarra; premiados en Alemania por el grupo Volkswagen en sus Premios a la Innovación por una aplicación instalada en una Tablet que permite reconocer el armario eléctrico enfocado y mostrar en pantalla las etiquetas virtuales que identifican cada uno de los elementos.

Con todo esto último, se intuye que iAR se dedica al sector de ingeniería industrial y/o mecánica, más en concreto, al desarrollo de soluciones industriales en Realidad Aumentada, Sistemas de gestión de mantenimiento, Monitorización y control de procesos, Movilidad industrial, Visión artificial, Realidad virtual, Realidad Aumentada y Aplicaciones web. Los proyectos que se desarrollan son de software industrial a medida, utilizando tecnologías avanzadas, como las nombradas anteriormente. Las áreas que abarca son el control y operación de equipos y maquinas, gestión interactiva de documentación técnica, gestión del mantenimiento, optimización de procesos y, en general, cualquier actividad de mejora industrial.

Las soluciones que desarrollan permiten eliminar el 100% del papel a través de gestores documentales, checklist digitales y sistemas de gestión de procesos, que se integran con las soluciones ya existentes en la empresa. Este va a ser el caso de estas prácticas, en las que se va a desarrollar un sistema integral de toda la gestión de producción para un cliente dedicado a la fabricación de perfiles metálicos, así como la gestión de todos los usuarios que harán uso de este sistema.

iAR provee un servicio de diseño y desarrollo de software que abarca desde la captación de los requisitos y el apoyo en la definición funcional, hasta su implantación y despliegue. Incluyendo el análisis, diseño y mantenimiento. Para obtener altos estándares de calidad, se utilizan las metodologías ágiles, más conocidas como SCRUM, dentro de cada una de las fases del ciclo de vida del software.

La empresa tiene una financiación privada y su sede se encuentra en Pamplona, Navarra. Cuenta actualmente con 35 empleados.

Problema a resolver

El Trabajo de Fin de Grado desarrollado resulta de un proyecto planificado por la empresa en la que se han realizado las prácticas curriculares desde febrero de 2021, presentada en el apartado anterior. Como ya se ha visto, iAR está dedicada al desarrollo de soluciones digitales en el entorno industrial, cuyo objetivo es el de obtener la información en el lugar y el momento que se necesita a través del dato único, facilitando las tareas del día a día.

El problema surge cuando una empresa dedicada a la producción industrial de perfiles metálicos abre una nueva nave. Hasta ahora, el flujo de información que se utilizaba para gestionar los procesos industriales de dicha empresa se realizaba a papel, por lo que se necesita la digitalización de todo el proceso de gestión del material que va a recibir, se va a tratar, y también se va a exportar de dicha nave. Ademas de esto, crear un sistema digitalización industrial de este calibre, obliga a crear otros sistemas que administrán todo lo referente a las cuentas asociadas a cada empleado, en el caso de que sea necesario utilizar el producto que se está por describir. Por tanto, dicha empresa acude a iAR, y se propone el proyecto del que se va a hablar en este Trabajo de Fin de Grado.

A partir de ahora, se mantendrá la privacidad de la empresa dedicada a la producción de perfiles metálicos por expreso deseo de iAR. Por lo tanto, para referirme a dicha empresa, me referiré a ella como cliente o como empresa contratante.

Nótese que se va a proceder a la descripción no solo de cómo va a ser la web, sino de todo el ciclo de vida del proyecto, el cual estará compuesto por: análisis, desarrollo y puesta en marcha.

Estado del arte

Las aplicaciones de la empresa se basan en la idea conceptual de que sólo con mirar o enfocar a un equipo con un móvil o Tablet, se obtiene toda su información asociada. A partir de este concepto, se están desarrollando aplicaciones relacionadas con la gestión y acceso a la documentación, checklist digitales, identificación de componentes, repuestos, órdenes de trabajo, asistencia técnica remota... Los productos que iAR crea se utilizan en el sector eólico, automoción, energía y fabricantes de maquinaria principalmente.

La empresa cuenta con el apoyo de varios clientes extranjeros, como son Alemania o Francia. Actualmente, el proyecto que se ha desarrollado en estas prácticas está destinado para unos clientes asociados de Austria.

iAR está asociada a dos empresas dedicadas igualmente a la digitalización industrial. La primera es Smart Lean Solutions [1], la cual tiene la sede en las mismas oficinas que iAR. Esta se dedica al desarrollo de soluciones de digitalización de las herramientas Lean Manufacturing, un método de organización del trabajo que se centra en la continua mejora y optimización del sistema de producción, eliminando desperdicios y actividades que no suman

ningún tipo de valor al proceso. La segunda empresa es Bambrai Artificial Intelligence S.L. Bambrai [2] está enfocada más a la Inteligencia Artificial; concretamente, a la industria, sector en el que cuenta con una amplia experiencia en la digitalización, toma de datos y tratamiento por medio de módulos de Machine Learning.

Sin embargo, aquí en Navarra, estas no son las únicas empresas que se dedican a la digitalización del sector industrial [3]. A continuación, veremos una lista de competidores dedicados a la digitalización de la industria:

- **Embeblue**: Ingeniería electrónica dedicada al desarrollo de dispositivos para el Internet de las Cosas e Industria 4.0.
- **Acuarela Digital**: Diseño web a medida y al Marketing online. Servicio integral, abordando la consultoría, legalidad web, servicio de hosting y dominios, comercio electrónico, desarrollo de aplicaciones, etc.
- **AVANCE software**: Empresa especializada en el desarrollo e implantación de aplicaciones de gestión integral ERP, producción y rentabilidad, especialmente diseñadas para Pymes y Micro-pymes.
- **Configuradores de producto 3D, S.L.**: Especializados en el diseño y desarrollo de herramientas de software dirigidas a mejorar y automatizar el diseño, presentación, venta y gestión de productos. Sus configuradores utilizan la tecnología 3D para ayudar en la comprensión y selección de productos con múltiples opciones.
- **Consultoria Informatica AD - HOC**: Especializada en ofrecer soluciones informáticas de desarrollo de software e implantación de sistemas a medida.
- **GM Vending S.A.**: empresa especializada en la fabricación y desarrollo de máquinas y software aplicado a soluciones atendidas y autoservicio en los puntos de venta (máquinas de vending de tabaco, dispensadores o armarios automatizados para retail, software y telemetría, dispensadores multiproducto, kioscos…).

Estos últimos son unos pocos de los competidores de iAR. Por lo tanto, la pregunta es, ¿existe algún producto que desempeñe la misma función de lo que se va a crear aquí?

Concretamente, el sector industrial en Navarra tiene un peso relativo muy significativo. Según datos publicados por el IEN el sector industrial, energía incluida, representaba en Navarra, en 2015, el 31,68% 1 del PIB regional. Esta aportación está muy por encima de los niveles medios de las economías española y europea situadas en un 16,4% 2 del PIB estatal y un 19,1% 3 del PIB de la UE respectivamente. Este sector reúne actualmente a 3.743 empresas y da empleo a 63.665 personas según los últimos datos publicados por el IEN4 correspondientes a 2015 [4].

Viendo el listado de empresas navarras dedicadas a la industria, nos damos cuenta de que, es probable que sí exista en el mercado industrial un sistema enfocado a la gestión de producción. A continuación, se presentan algunas de las soluciones que se enfocan en la gestión de producción, similares a la que se ha realizado en este proyecto:

- **Nexus Integra**: es la plataforma de integración de sistemas industriales abierta y estándar capaz de generar un entorno global de operaciones y monitorización para la gestión de activos industriales a gran escala. Consigue gestionar todos los datos generados por las empresas y los muestra de la forma más intuitiva [5].

- **NutSL**: dispone de varias soluciones dentro del marco de la Industria 4.0, como son: [6]

    - NUT-DUPLA: desarrollo e implantación de sistemas MES para la toma de datos en planta, especializados en el sector de la alimentación.
    - NUT-GMAO: una solución totalmente integrada en Microsoft Dynamics NAV que ayuda a informatizar y gestionar las tareas de mantenimiento.
    - NUT-LIMS: software de gestión de laboratorios que permite planificar, identificar y clasificar, muestras, y a partir de las mismas planificar y ejecutar ensayos y análisis, para finalmente emitir los informes que se requieran.

Por lo que, ¿porqué iAR para desarrollar esta solución? Esto es debido a que iAR ya había desarrollado anteriormente productos parecidos para esta empresa, sin embargo, con el motivo de la apertura de la nueva nave, el cliente necesita innovar. Además, iAR es conocido por ser pionero en soluciones innovadoras, donde implementan herramientas poco convencionales, que son muy eficientes.

Mi participación en iAR

Para este proyecto, me he incorporado al equipo de FrontEnd, dedicado a la parte cliente, es decir, a desarrollar la interfaz de usuario, con todas sus vistas y funcionalidades. También se dedica a hacer todas las llamadas Web Service al servidor de Back End, y de gestionar todas las respuestas, volcando la información en la página web Microsoft Office

El aspecto a recalcar del método de trabajo en este proyecto ha sido el del uso de la metodología ágil o SCRUM. Usando SCRUM, el avance incremental del proyecto se ha dividido en iteraciones, que marcan cómo y en qué grado ha progresado este. En iAR, cada iteración es una semana. Al concluir la semana, las tareas y el tiempo que se ha dedicado en estas debe quedar registrado.

Para la realización de este proyecto, se me proporcionó un ordenador Mac. El modelo del ordenador Mac es MacBook Air (Retina, 13 pulgadas, 2019). Cuenta con un procesador 1,6 GHz Intel Core i5, una memoria RAM de 8 GB 2133 MHz LPDDR3, los gráficos integrados en el

procesador Intel UHD Graphics 617 1536 MB, una memoria interna de 256 GB y un sistema operativo macOS Mojave 10.14.6.

La web que se ha creado es Responsive, es decir, válida para cualquier tipo de dispositivo, debido a que se va a utilizar tanto en escritorio como en Tablet. Por esto se me proporcionó también una Tablet Samsung Galaxy Tab A 2016. Con esta se han realizado las pruebas, además de comprobar que la web cumplía con los estándares Responsive.

Aparte de esto, me han dado todos los tutoriales y formaciones necesarias para poder desempeñar las prácticas de una manera correcta. Además, el resto de la plantilla ha mostrado mucha disponibilidad en el caso de que me surgiese alguna duda. Desde el primer día se me hizo una cuenta propia de la empresa, con las licencias necesarias para poder utilizar todas las herramientas de Microsoft Office.

Análisis y diseño del sistema

Análisis

Para este proyecto, y otros realizados en iAR, se utiliza el análisis DAFO, donde se han considerado las oportunidades que presenta la realización de este proyecto para iAR. Contextualizando, hay que definir qué es un análisis DAFO y para qué se utiliza. DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades), es una sencilla herramienta de análisis estratégico muy extendida en la toma de decisiones de todo tipo de organizaciones y empresas. Por tanto, resulta de gran utilidad para emprendedores, autónomos y pymes a la hora de analizar sus proyectos y negocio, así como elaborar sus estudios de mercado, planes estratégicos y de negocio. En el caso de iAR, esta se vale de este tipo de análisis, sobre todo para considerar qué empresas van a aprovechar todo su potencial, y qué proyectos merece la pena llevar a cabo.

Sabiendo esto, iAR tiene muy presente las fortalezas que posee, como son mayoritariamente las soluciones digitales en el mercado utilizando técnicas como la realidad aumentada. Sin duda, iAR ha sido pionera en utilizar estas técnicas, lo que ha servido para darse a conocer en la industria navarra. De esto se sirve iAR para mostrar a las industrias sus capacidades de resolución.

Otro aspecto que se debe considerar es el afrontamiento de una propuesta de proyecto, es decir, las oportunidades que se presentan. Para esto hay que tener en cuenta muchas cosas, inicialmente como va a repercutir en el mundo laboral la realización de ese proyecto para iAR. El equipo comercial se ha reunido con el cliente para identificar sus necesidades. Si es necesario, se apoya en el equipo técnico de iAR que le da soporte en el proceso. El objetivo es conseguir una venta, mediante la aceptación de una oferta.

Para considerar una oportunidad o propuesta válida, antes se han elaborado una serie de documentos, que hicieron posible considerar esta propuesta, y después, han ayudado a la realización de la misma. En primer lugar, vemos la especificación de alto nivel, donde se han redactadodos documentos, el DDF (Documento de Definición Funcional), entendible a nivel de usuario y elaborado por el equipo comercial, y el DDTF (Documento de Definición Técnica y Funcional), donde se presentan todos los aspectos técnicos a tener en cuenta. Al mismo tiempo se redactan otros tres documentos: el cronograma, donde se explica la planificación del calendario del proyecto a grandes rasgos, las estimaciones, donde se recogen las estimaciones de las horas que se han contemplado a dedicar en un proyecto, y el presupuesto, que recoge la oferta económica del proyecto.

Finalmente, una vez elaborados los documentos anteriores, toda la información se recoge en un archivo final, DEAN, donde se recogerán todas las horas estimadas por bloques, con sus respectivos presupuestos y sus cargas de trabajo, que servirá de referencia también para el cliente.

Objetivos

Visto lo que se ha explicado anteriormente, el objetivo de este proyecto es el de desarrollar un conjunto centralizado de web's. La empresa cliente administrará los datos iniciales, como son los pedidos de compra y los pedidos de clientes con SAP, un software de planificación de recursos empresariales. Aparte de eso, iAR estará encargado de crear un producto con una interfaz que permita al usuario:

- Para administración:
    - Visualizar todos los usuarios de sistema
    - Crear, eliminar o modificar las cuentas de los usuarios
    - Visualizar todas las zonas existentes
    - Crear, eliminar o modificar zonas
    - Asociar usuarios con zonas
    - Crear checklist
    - Gestionar todas las entregas de los proveedores
    - Visualizar y gestionar todo lo referente a los pedidos de compra
    - Visualizar y gestionar todo lo referente a las órdenes de trabajo (OT) creadas en producción
    - Visualizar y gestionar todo lo referente a los envíos al cliente

- Para los operarios:
    - Registrar todas las entregas recibidas de materia prima
    - Situar el material en los almacenes
    - Crear y organizar las OT's de las que se compone la producción
    - Registrar todo lo referente a las OT's
    - Registrar de todos los paquetes
    - Registrar todo lo referente a los envíos

Estos requisitos han sido definidos por la empresa cliente y el jefe de equipo de desarrollo de iAR, el cual les ha guiado respecto a los detalles técnicos.

Patrón de diseño

En nuestro proyecto, como en todos los que diseña y desarrolla iAR, se utilizará el patrón de diseño MVC. Un patrón de diseño software es una solución general y reutilizable aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales de un proyecto.

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. En este proyecto Web, se ha implementado esta metodología que facilita mucho la organización de un proyecto. Para cada uno de los módulos y pantallas de las cuatro webs, existe una división entre los datos, la vista y el controlador.

Como ya se ha dicho, el objetivo general es separar los componentes de un módulo. Las vistas son plantillas o templates, que consisten en los archivos PHP o HTML de cada uno de los componentes de las webs.

El modelo consiste en los componentes encargados de obtener los datos necesarios para cada uno de los módulos de las webs. Consisten en archivos JavaScript, donde se hacen peticiones Ajax, las cuales se explicarán detalladamente más adelante.

Por último, el controlador de cada uno de los componentes son archivos JavaScript, que recojan los datos de sus modelos correspondientes y los vuelquen en las vistas.

Diseño WEB

DDFT

El DDFT (Documento de Definición Técnica y Funcional) describe todas las funcionalidades técnicas que va a tener el producto final. Para generalizar, lo que se ha hecho es un conjunto de páginas Web Responsive, es decir, que se acople a cualquier dispositivo, sea cual sea la resolución de su pantalla.

Se han creado en total 5 Web's, las cuales son Suite, Identity, BluePrint, Production y Checklist. Para contextualizar, cada usuario trabajará con una cuenta, y dependiendo del perfil que tenga, podrá hacer un número limitado de cosas. La Suite se ha creado sencillamente para acceder a las demás Web's desde un solo sitio. Identity se encargará de la administración y gestión de todo lo referente a las cuentas de usuarios. Blueprint y Production se encargará de tramitar el proceso de producción y la información de esta. Por último, está Checklist, la cual se ha replicado de la ya existente web propia de iAR [4].

El proyecto se basa en el estilo de arquitectura modelo-vista-controlador. Esta separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, lo cual ofrece una mayor organización y un flujo de datos mucho mejor definidos.

Login

El Login no es una web como tal. Consiste en uno de los módulos que se encuentran en la web de Identity, de manera que desde cualquier otra página se pueda acceder a él. Como se explicará más adelante, el servidor de Back que se va a utilizar provee cuatro servicios

diferentes, uno para el identity, otro para el blueprint, otro para la producción, y el último para el checklist.

La idea es que, para acceder a cualquiera de estos servicios, antes uno debe pasar por el Login y acreditar su identidad en estos cuatro servicios. Esto es algo que se hace con el uso de cookies y tokens, que se explicarán más adelante. El Login deberá ser una página plana, que tenga simplemente el logo de la empresa cliente, y dos campos, uno para el alias del usuario que desea acceder a la Web, y la contraseña que coincida con la cuenta de dicho usuario.

En el caso de que el usuario no exista, se lanzará una alerta, con los estilos predeterminados del navegador en el que estemos, informando del error cometido. Si la contraseña no coincide, se lanzará igualmente una alerta, con un mensaje avisándonos de que la contraseña no coincide. Existe otro caso, en el que el Login de uno de los servicios haya fallado, por lo que se abrirá la Web, sin embargo, las funcionalidades que son propias del servicio cuyo Login ha fallado no deben mostrarse. De igual manera en este último caso se abrirán tantas alertas como servicios hayan fallado. En la Figura 1, se muestra el mockup inicial del módulo de Login.

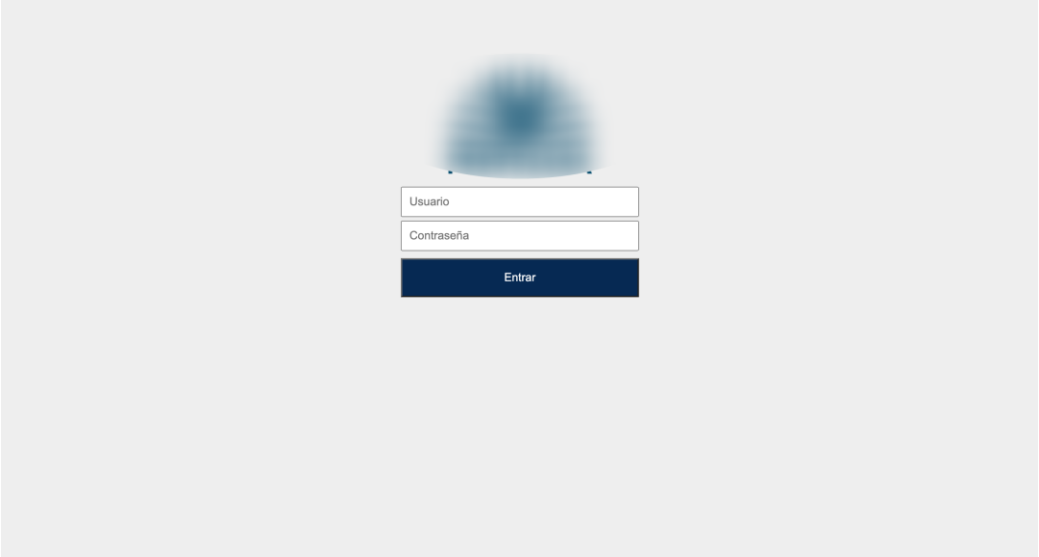


Figura 1. Mockup Login

El Login se muestra cuando se comprueba que no existe ningún token en la memoria local del navegador, es decir, cuando nadie haya hecho el Login, o cuando una sesión haya expirado. La idea es cuando se acceda a cualquiera de las web's si no existe este token, que aparezca esta pantalla, de manera que una vez el usuario se haya identificado con sus credenciales, se abra la página deseada. Más adelante se mostrará el flujo de las webs.

Logout

Al igual que el Login, el Logout es un módulo propio de Identity. Cuando en cada una de las webs se pulse el botón de Logout de la cabecera, se mostrará el contenido de Logout, que

consistirá en una simple pantalla de carga. Este módulo hace las operaciones suficientes para eliminar el token guardado. Una vez a eliminado el token, abrirá la pantalla del Login para empezar en el punto de partida. Esta pantalla de carga es la que se utiliza en el resto de las webs cuando se está cargando un proceso para mostrar que las acciones se están desarrollando.

Suite

La idea es crear una web principal, donde se centralizan todas las acciones de la empresa cliente. Para ello se ha creado la web Suite. Esta consiste simplemente en un menú, con las cuatro opciones que se han desarrollado en este proyecto: Identity, Blueprint, Production y Checklist. En el caso de que se pulse una de estas opciones, se deberá abrir otra página en la misma pestaña correspondiente al item que se ha pulsado. Es decir, si se pulsa la opción Identity, se mostrará la página de identity. Inicialmente, se propuso la idea de mostrar en el menú algunas otras opciones que, que principalmente pertenecían a la web de producción, sin embargo, se descartó la idea, y se dejaron las opciones originales.

En la página se muestra un header, es decir, una cabecera, donde se muestra el logo de la empresa cliente, el título de la página, que en este caso no tiene, y un menú desplegable, que nos dará las opciones de cada una de las webs y de hacer Logout, es decir, de eliminar el token existente. En el menú principal, es decir el contenido de la página, se mostrará aquellas opciones a las que tenga permiso de acceso el usuario que ha entrado. En el caso de los perfiles encontramos tres tipos, administrador, oficina y operario. Los usuarios con rol de administrador tendrán acceso a cualquiera de las webs desarrolladas. Oficina tiene acceso a todas las opciones menos a la de Identity. Por último, los usuarios con rol de operario tienen acceso únicamente a las opciones de Blueprint y Production, es decir, aquellas relacionadas con la producción.

El estilo de las opciones del menú serán recuadros del color principal de la empresa cliente. En cada uno de ellos se mostrará el nombre de la web, junto a un icono representativo. En la Figura 2 se muestra cual ha sido el resultado de la página.

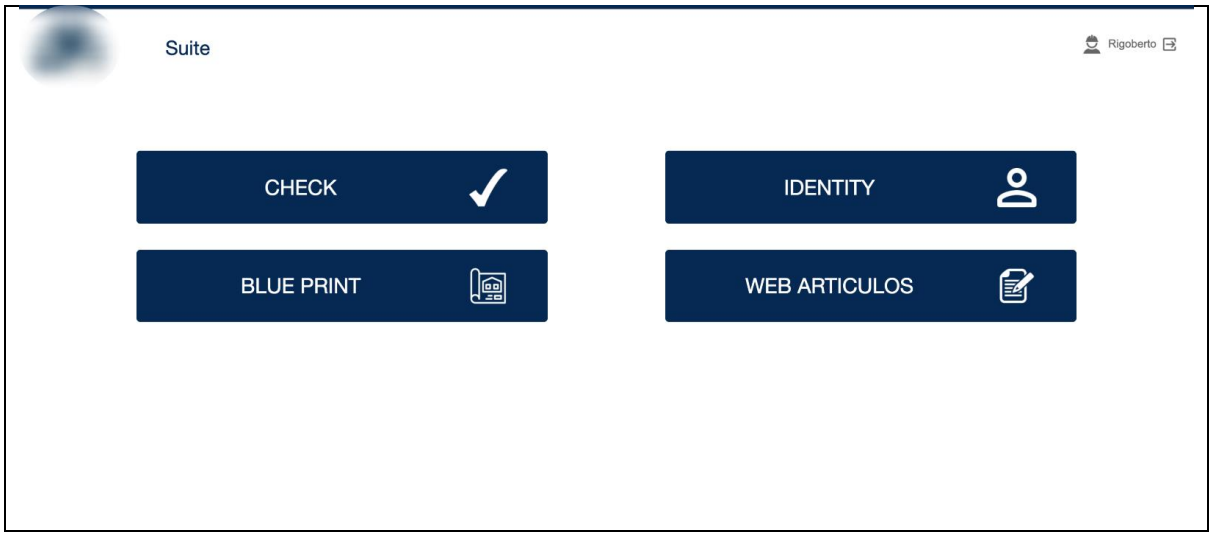


Figura 2. Mockup Suite

Identity

Como ya se ha dicho, a esta web solo tienen acceso los usuarios con cuyo perfil sea administrador. Esta va a ser la encargada de administrar todo lo relativo a las cuentas de los usuarios. Cada usuario va asociado a un alias, que será único. El Uid es otro atributo único e intransferible de cada cuenta, sin embargo, este irá por detrás de la vista, y el usuario utilizando identity no verá nada relativo a este.

Se nos da la posibilidad de crear un usuario. De este, se introducen el nombre, apellido, el rol que se le va a dar, es decir, su perfil, el mail y el alias asociados a la cuenta, los cuales no tienen por qué ser el mismo, y por último la contraseña y su confirmación. Evidentemente, la confirmación de la contraseña es un último campo donde se tiene que volver a introducir la contraseña deseada. Si alguno de los campos está vacío, el alias ya existe, o las contraseñas no coinciden, se debe de abrir una alerta, informando del mensaje de error.

En la web, la cual tiene la misma cabecera que Suite, con diferente título, se muestran el listado de todos los usuarios del sistema gestor. Se pueden filtrar la búsqueda de usuarios en función del nombre, del alias o del rol. Cada fila del listado muestra dos botones, uno para modificar o editar al usuario, y otro para eliminarlo. Esa es la razón principal por la que solo un usuario con perfil administrador puede acceder a esta web. Evidentemente, el usuario que esté utilizando la web no puede eliminarse a sí mismo. A la hora de modificar un usuario, se puede cambiar únicamente su nombre, apellido, mail, rol o perfil y contraseña. El alias no se puede modificar, siempre será el mismo para el usuario. Cuando se elimina un usuario, el alias que tenía asignado ya está disponible para un nuevo usuario.

Los estilos de la página coinciden con el diseño de las otras webs. El mockup de Identity se muestra en la siguiente Figura 3.

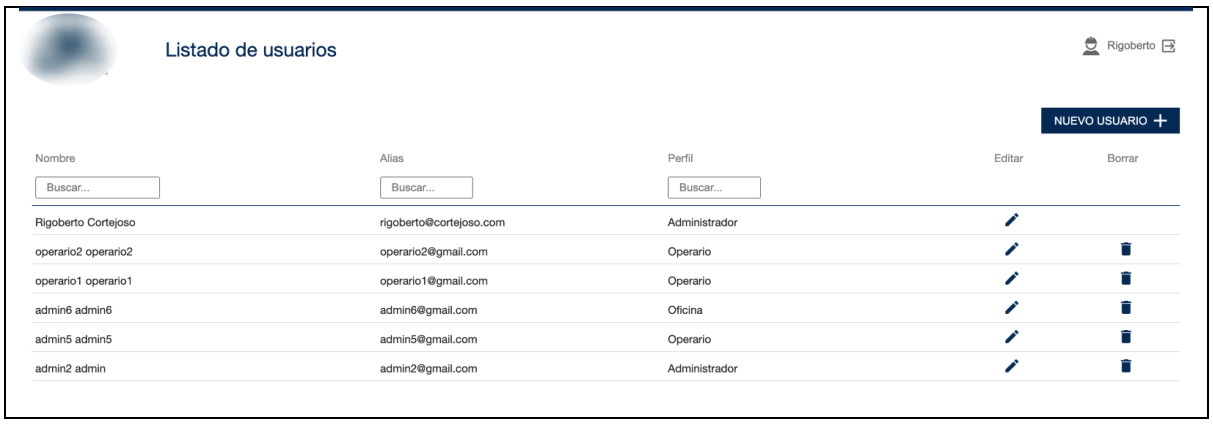


Figura 3. Mockup Identity

BluePrint

Blueprint es la web en la que se llevan a cabo todas las tareas relacionadas con las zonas. Estas son las zonas de trabajo de la empresa cliente, tanto las entradas y salidas de material en la producción, como todas las líneas de fabricación, las cuales aún están pendientes de definir por la empresa cliente. Cada una de las zonas tiene que tener asociado un Uid, es decir, un identificador que el usuario no ve, el nombre, que también será único, y una descripción. A las zonas habrá asociados un número limitado de trabajadores, por lo que, a la hora de diseñar la base de datos, hemos necesitado una relación entre zonas y usuarios.

Tenemos que tener la posibilidad de crear una zona; cuando se pulse un botón, se abrirá un modal, donde tendremos que introducir el nombre y la descripción de la zona. Antes de crear la zona, se comprueba que el nombre introducido ya esté asignado a otra zona existente, y si es el caso afirmativo, se abrirá una alerta avisándonos del error.

También se muestra un listado de todas las zonas existentes, y para cada una de las zonas existen diferentes posibilidades. En primer lugar, tenemos un botón que nos elimina la zona, y, por tanto, todas las relaciones que existían entre esa zona y los usuarios asociados a ella. Antes de confirmar esa acción, se abre una alerta que nos avise de que la acción es irreversible. También, pulsando el nombre o la descripción de la zona, se abre el mismo modal que se utiliza para crear una zona nueva, sin embargo, los campos del nombre y descripción no están vacíos, si no que contienen los valores correspondientes de la zona seleccionada. Se cambiarán los malores anteriores a los que se desean. Una vez se dé en el botón de “Guardar Zona” del modal, se confirmará la operación.

De cada zona también tendremos la posibilidad de acceder a la web del Checklist, para crear formularios propios de esa zona. En la Figura 4, se nos muestra la vista que tiene la web Blueprint.

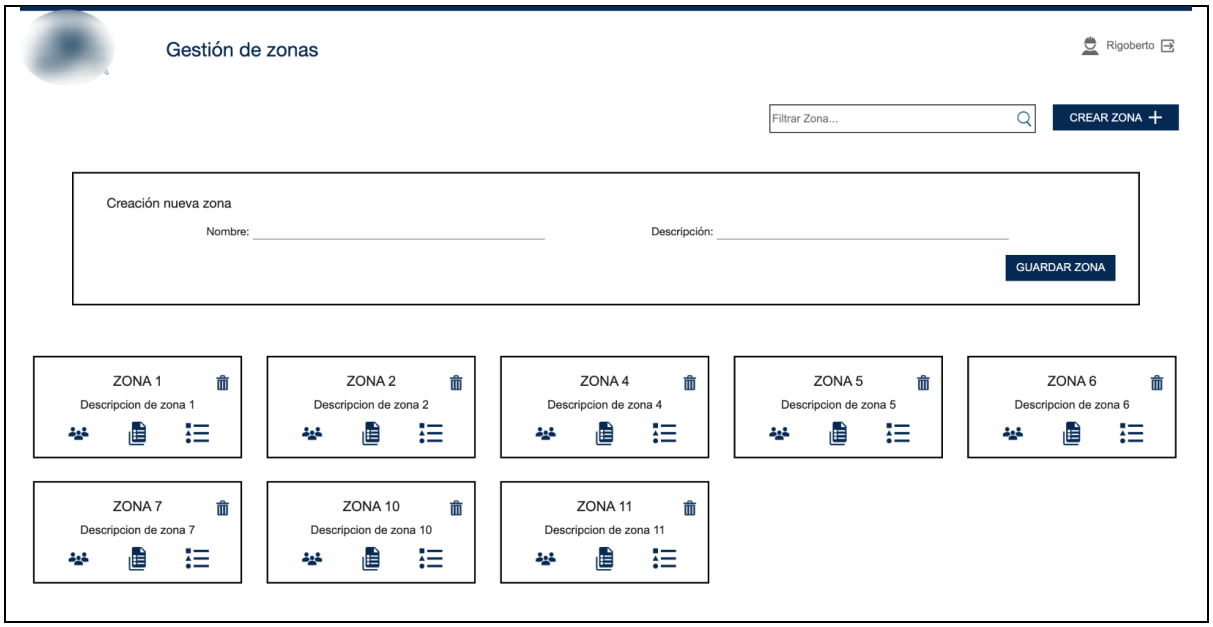


Figura 4. Mockup Blueprint

De cada zona, también tenemos un botón, el cual siendo pulsado nos abrirá el contenido del módulo UserList. Al igual que la Suite tiene diferentes módulos, como Login o Logout, Identity tiene su propio módulo, el cual no es utilizado por el propio Identity, sino que se usará en la web de Blueprint. Este módulo es UserList.

Este consiste en una web que muestra un listado de usuarios, todos los usuarios del sistema. A la página se le pasará por parámetro en la url el Uid de la zona a la que queremos asociar los usuarios que queramos. Cada usuario tiene un checkbox, el cual estará asignado en el caso de que ese usuario esté asignado a la zona en cuestión. Debajo de la lista de usuarios, hay un botón “Asignar”, que, al ser pulsado, añadirá los usuarios que se han seleccionado esa vez a la zona. En el caso de que un usuario ya seleccionado se quede como está, entonces cuando se pulse asignar no se tiene que hacer nada. Si un usuario estaba asignado, sin embargo, eliminamos su selección, entonces ese usuario se eliminará de la lista de usuarios asignados a la zona.

De cada usuario se muestra un checkbox, cuya función ya está explicada, el Uid del usuario, su nombre, apellido, y su perfil de usuario. En la Figura 5 se muestra como es la página de asociación de usuarios a una zona designada.

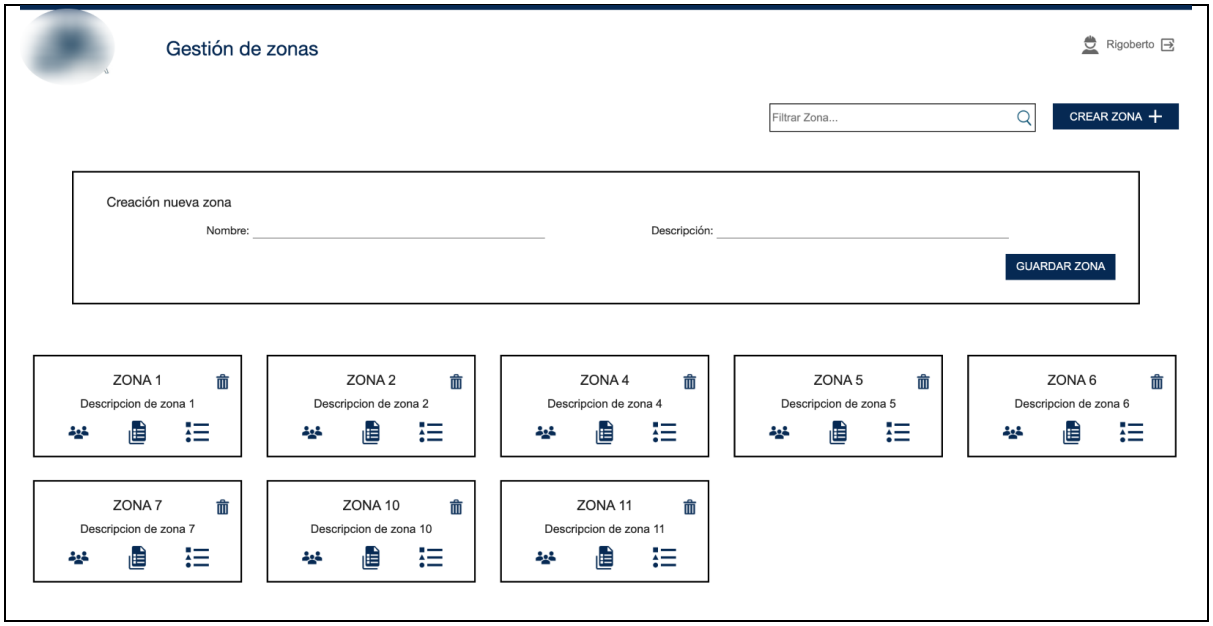


Figura 5. Mockup UserList

La web de Blueprint se utilizará en el proceso de producción de la empresa cliente, por lo que a esta tienen acceso únicamente aquellos usuarios que tengan el perfil de administrador o de operario.

Production

Para la producción de perfiles metálicos, se definen tres fases que componen todo el proceso de fabricación: Entradas, Fabricación y Expediciones o Envíos. Los datos iniciales que se usan para empezar todo el proceso de producción se extraen de la base de datos personal de la empresa cliente, que son los pedidos de compra, es decir, los pedidos de materia prima que se hacen al proveedor, y los pedidos de cliente, que consisten en todos los encargos que los socios le hacen a la empresa cliente de los perfiles deseados. A continuación, se describirán cada una de las fases al detalle.

Production - Entradas

En primer lugar, para la parte de los operarios, tenemos las entradas. La entrada de material se trata de la recepción de material a la fábrica. Cuando se recibe el material el operario deberá crear una entrada. En los documentos que trae el transportista aparecerán dos códigos de barras, uno con el número de pedido y otro con el albarán de esa entrada. El operario puede ver el listado de entradas, junto a su información, como se ve en la Figura 6.

Figura 6. Mockup Entradas

El operario deberá leer esos dos códigos, el primero estará relacionado con un pedido de entrada de SAP. Cuando se asocie con dicho pedido ya tendremos toda la información almacenada en SAP cargada en la pantalla para que el operario pueda registrar todas las bobinas que lleguen. La bobina es la unidad de material propia de la empresa cliente. Se trata de unas láminas de longitud y grosor variable, enrolladas formando un cilindro, que se unirá a la devanadora para así comenzar la fabricación.

El segundo código de barras, el número de albarán, simplemente se leerá para asociar ese número de albarán con ese pedido. El transportista también proporcionará al operario un certificado de calidad, el cual habrá que fotografiar. La pantalla resultante se muestra en la Figura 7.

Por lo tanto, para generar la entrada será necesario:

- Leer el código de barras con el número de pedido proporcionado por el transportista.
- Leer el código de barras con el número de albarán proporcionado por el transportista.
- Hacer una foto al certificado de calidad.
- Hacer una foto al albarán.

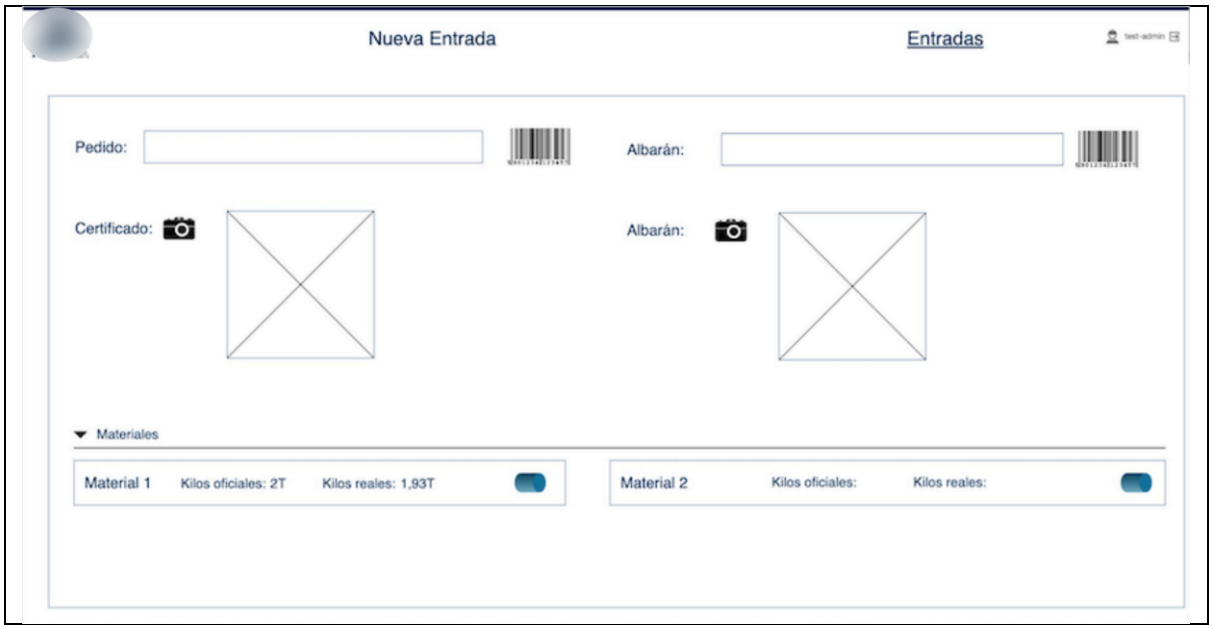


Figura 7. Mockup Nueva Entrada

Se elegirán las bobinas de cada material, podemos ver tanbién un listado de todos los materiales, para así registrar las bobinas necesarias. Una vez introducido estos datos, el operario ya podrá registrar todas las bobinas. Para esto, es necesario que el operario:

- Indique el peso real de la bobina.
- Realice el checklist de recepción. Para realizar el checklist se redirigirá al iAR-Check y se asociará el UID del reporte con esta bobina. Este checklist estará asociado a la zona “entrada”.
- Imprima la etiqueta.

- Asigne la posición del almacén donde se almacenará la bobina.

Las posiciones del almacén estarán guardadas en la base de datos. En la Figura 8 se muestra la pantalla para el registro de todas bobinas.

Figura 7. Mockup Nueva Entrada

Desde la base de datos del cliente en SAP, se mostrará para la parte de administración, es decir, en la web para escritorio, un listado de todos los pedidos de compra. Es un listado similar al de entradas para la parte de los operarios. De cada pedido de compra se podrá ver el detalle en la pantalla de Detalle Pedido compra, mostrada en la Figura 8. En ella se encuentra el resumen de todas las bobinas registradas, con sus pesos, calidad, desarrollo, espesor, ubicación y reporte de verificación de entrada. Para acceder a este reporte y ver el resultado se accederá a la web del iAR-Check.

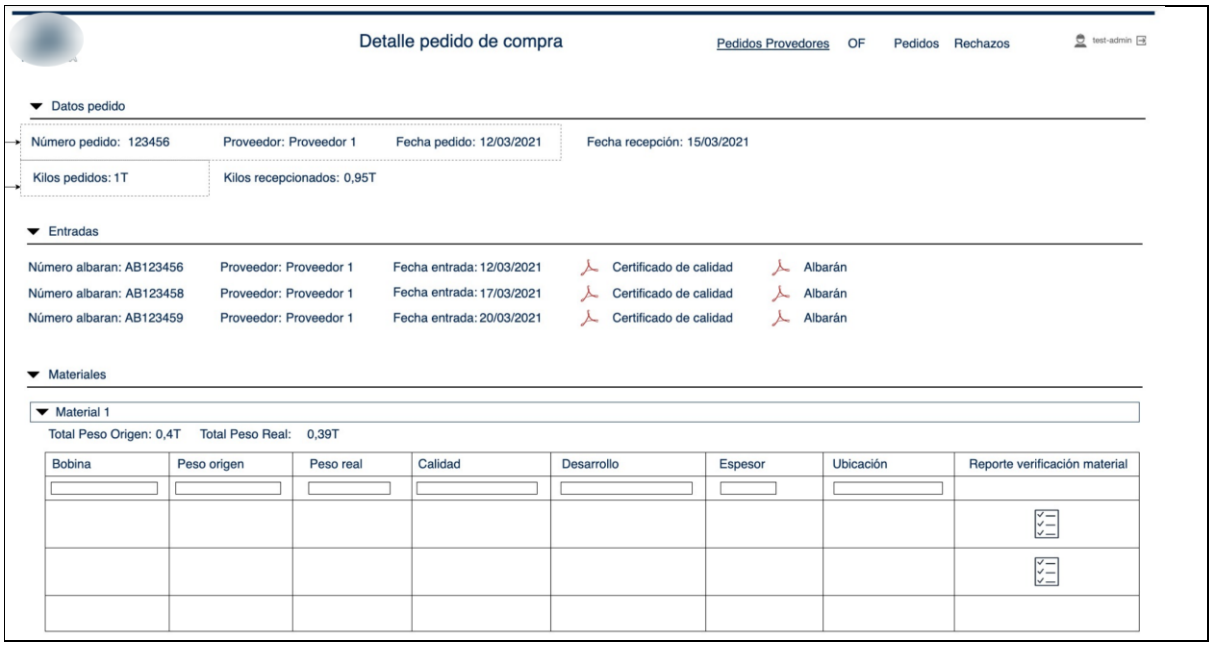


Figura 8. Mockup Detalle Pedido Compra

La idea es que dependiendo del rol del usuario que haya iniciado la sesión, pueda ver una parte de la web u otra. Es decir, si un usuario que vaya a trabajar en oficina tiene el rol de oficina, solo podrá ver la parte de administración; de la misma manera, un operario solo podrá ver la parte de las entradas. Estos Mockups muestran el contenido de las pantallas, las cuales al ser Responsive no es necesario preocuparnos de la resolución de las pantallas en la que se muestre la web, pues se adaptan a cualquier tamaño de pantalla.

Production - Fabricación

En segundo lugar, se va a proceder a explicar la segunda fase, la de fabricación. La unidad de trabajo para esta fase es la OT (Orden de Trabajo). Para el proceso de fabricación, se tendrá que hacer desde las oficinas la planificación de la producción a partir de pedidos de clientes almacenados en SAP. Para ello se deberá crear tantas OTs como sean necesarias para cada pedido. Se diferencian dos tipos de OTs:

- OT de fabricación (A partir de ahora OTF). Estas OT son las que están relacionadas con los pedidos de los clientes.
- OT de mantenimiento (A partir de ahora OTM). Estos OT son todas aquellas operaciones que se realicen en la línea que no sea fabricación, como, por ejemplo: cambio de utillaje, mantenimiento, limpieza, etc.

Para la creación de una OTF, serán necesarios los siguientes datos:

- Pedido del cliente, leído de SAP.
- Artículo a fabricar
- Cantidad a fabricar
- Fecha programada
- Unidades por paquete
- Línea de fabricación
- Periodicidad del checklist de calidad. Nos indicará cada cuantos paquetes o unidades, debe saltar un aviso para que el operario realice un checklist de calidad en lo fabricado.

Para la creación de una OTM, serán necesarios los siguientes datos:

- Tipo de OTM (cambio de utillaje, limpieza o mantenimiento)
- Fecha programada
- Línea de fabricación

Todas las OTs pueden tener los distintos estados que se muestran en la Figura 9. No puede tener más de un estado al mismo tiempo.

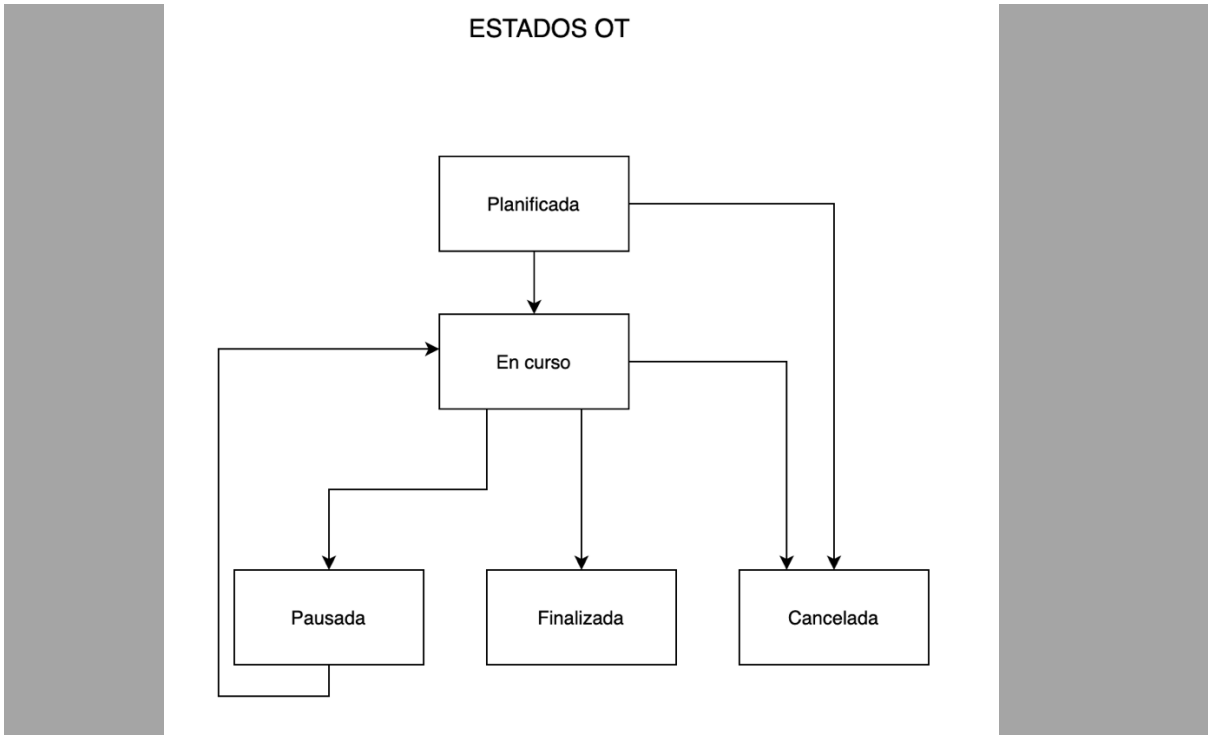
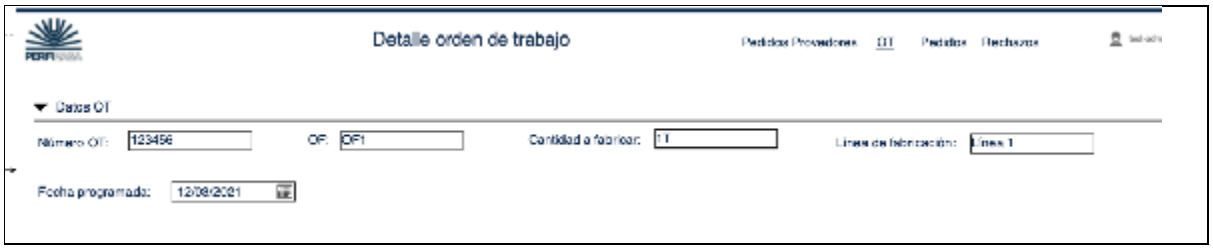


Figura 9. Estados OTs

Desde las oficinas podrán seguir el estado y toda la información de las OT en una pantalla parecida a la que muestra la Figura 10 donde también tendrán información de las bobinas consumidas en cada OT.



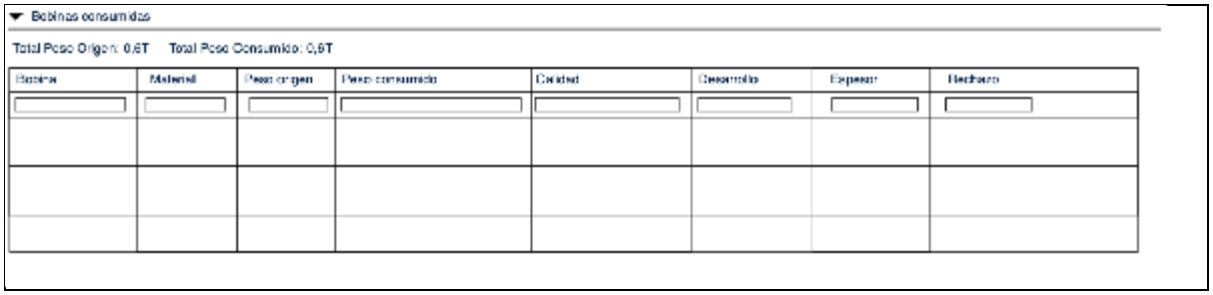


Figura 10. Mockup Detalle Pedido Compra

Una vez se ha planificado las OTs, los operarios ya tienen la planificación de lo que se debe realizar. Cada devanadora, primera máquina de inicio de la línea dispondrá de una Tablet donde podrá ver las OTs que hay planificadas. Cuando es necesario introducir una bobina en la devanadora, se pueden dar dos situaciones.

- Que la bobina esté preparada para entrar, en un puesto detrás de esta.
- Que la bobina siga en el almacén en la posición donde se ha guardado.

Por lo tanto, tenemos dos escenarios, para el primero será necesario realizar una preparación del material y para el segundo no. En ambos casos conociendo la OT que se va a fabricar el sistema debe sugerir al operario que bobinas coger del almacén de acuerdo con las características de la OT. Para esto se tendrá en cuenta el sistema FIFO, aunque en primera posición deberán aparecer los retales. Esto será a modo de información y le indicará al usuario en que posición del almacén se encuentran cada una de las bobinas o retales sugeridos.

Con esto, será el propio operario el que seleccione que bobina se va a utilizar para trabajar esa OT y no tiene por qué ser las que le ha sugerido el sistema. Para seleccionar la bobina, el operario deberá leer con una pistola lectora de códigos de barras, el código que tenga impreso en la etiqueta dicha bobina.

Una vez se ha seleccionado con que bobina trabajar, se debe dar su entrada en la devanadora, para ello se debe leer desde la Tablet de la devanadora el código de barras de la etiqueta y el sistema deberá comprobar que la bobina leída tiene las mismas características que la que sea necesaria para la fabricación de esa OT.

Cuando no se haya preparado la bobina para trabajar estos dos pasos, se realizarán desde la Tablet de la devanadora, por lo que será un único paso. Tras la entrada de la primera bobina, se podrán generar más entradas hasta la fabricación del artículo. A continuación, en la Figura 11 se muestra un ejemplo de pantalla para la devanadora.

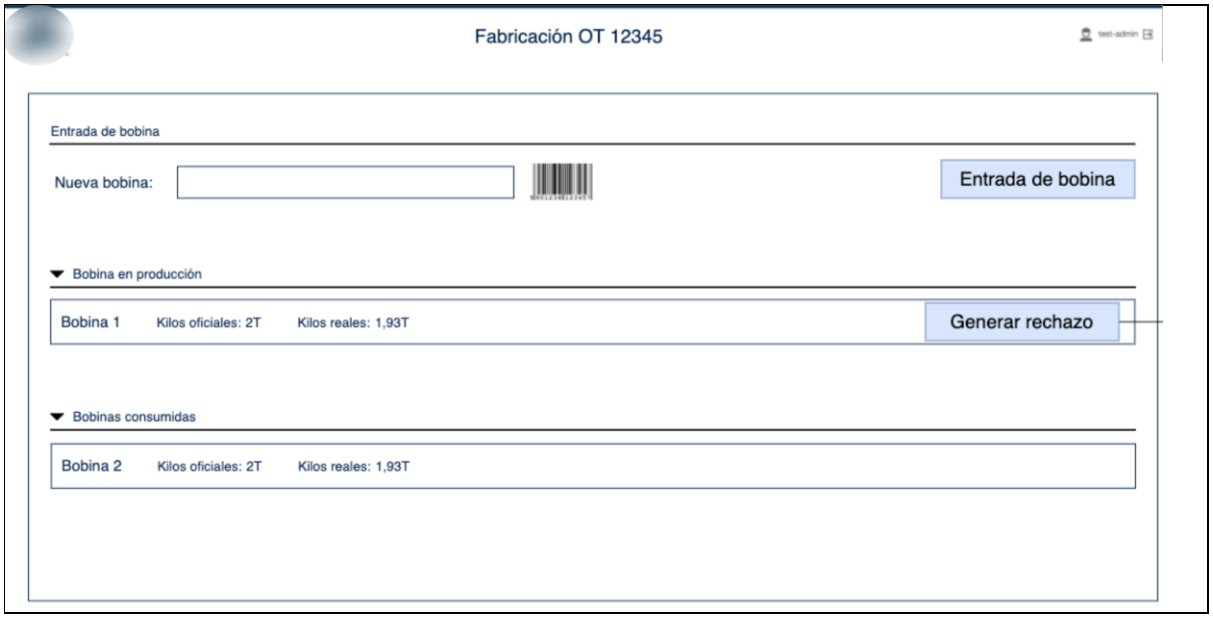


Figura 11. Mockup Fabricación OT

Estando trabajando con una bobina puede darse el caso, de que haya que retirar la bobina porque es un rechazo ya que tiene algún defecto, por eso aparece una opción para generar un rechazo de la bobina que se está utilizando para esa OT.

Otra de las acciones que puede realizar el operario es la de pausar la OT, esto conlleva que esa OT se queda en estado “pausada” y se comienza a fabricar una OT con la misma bobina que está en la línea o que se genera un retal de esa bobina.

Al final de línea tendremos otra Tablet para:

- Generar los paquetes de los productos fabricados
- Generar el rechazo de la bobina
- Cambiar bobina
- General retal
- Finalizar OT

En esta pantalla de final de línea se le irá informando al usuario cuantos paquetes ha generado de los que tiene que generar, que bobina está consumiendo y cuál es la siguiente bobina. Aquí aparecerá un aviso, una vez se hayan generado los paquetes indicados en la OT, para realizar el checklist de calidad. Este checklist se completará desde la web de iAR-Check. Al final de la línea se tendrá una pantalla parecida a la Figura 12.

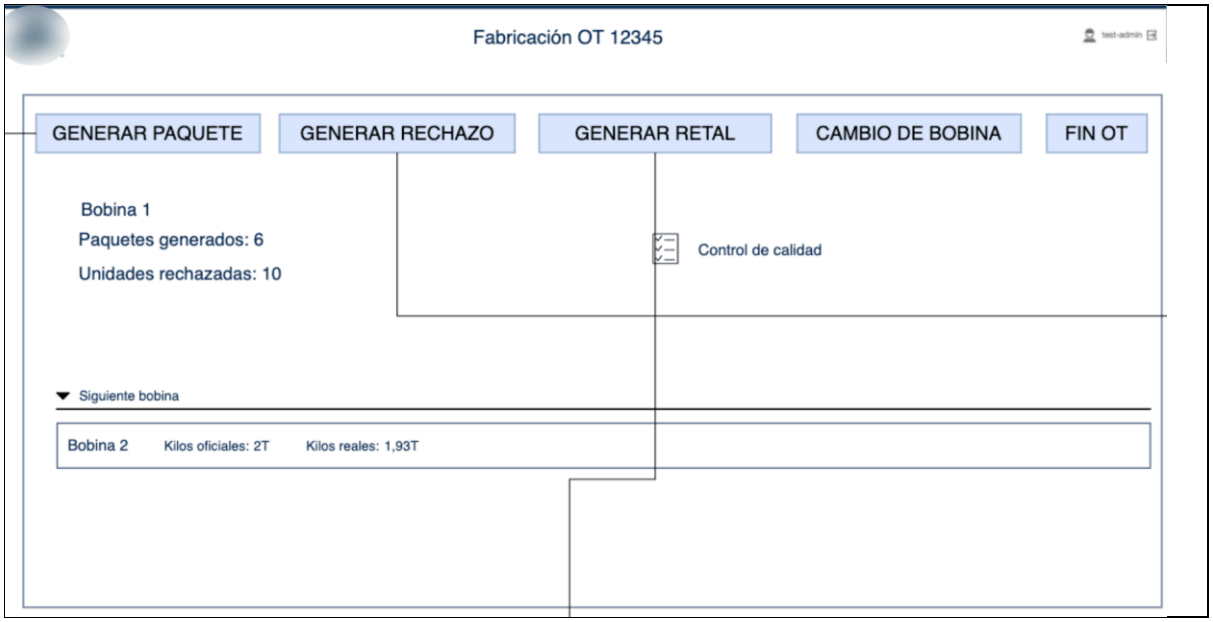


Figura 12. Mockup Fabricación OT Paquetes

Para la creación de paquete habrá que identificar el paquete, indicar cuantas piezas van en el paquete si no está completo y hacer una foto del paquete. Una vez esto, se imprimirá una etiqueta que deberá colocarse en el paquete. Tras identificar cada paquete habrá que realizar el flejado y posteriormente asignarle una ubicación del almacén, para ello se tendrá que leer el código de la ubicación para realizar la asociación.

Para generar un retal, se necesita:

- Asignarle un nuevo peso a la bobina.
- Indicar que ahora la bobina es un retal.
- Generar etiqueta, para etiquetar el retal.
- Asignarle ubicación al retal.

Para generar un rechazo, se necesita:

- Asignarle motivo de rechazo.
- Generar etiqueta, para etiquetar el rechazo.

Tras lo descrito anteriormente las bobinas tendrán los siguientes estados:

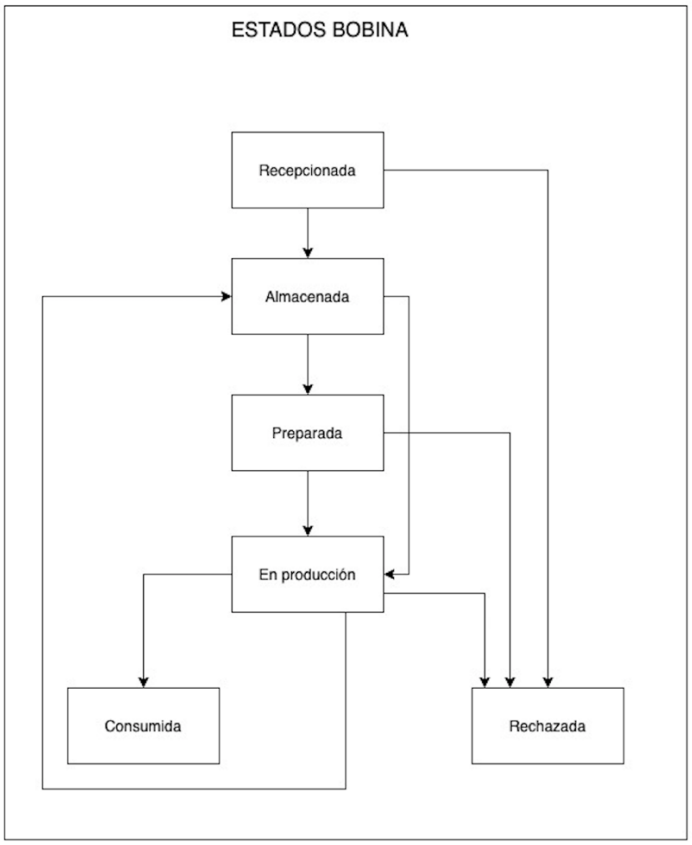


Figura 13. Estados Bobina

Y los estados de los paquetes:

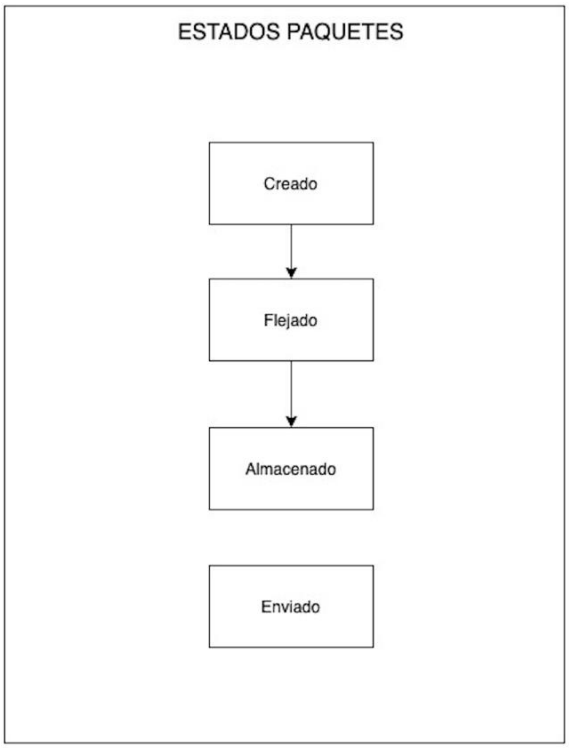


Figura 14. Estados Paquete

Production - Expediciones

En último lugar, están los envíos de los paquetes generados. Para realizar la expedición se han de seguir los siguientes pasos:

- Primero desde oficina se tiene que crear una orden de carga (OC), que también se tendrá que dar la opción de que esta orden de carga la genere el propio operario que va a realizar la expedición.
- Creación del packing-list
- Realización de checklist de verificación de envío
- Generación de albarán
- Generación de carta de porte
- Generación de informe con toda la información anterior para cliente final.

Una vez el operario tiene creada la OC, debe buscar los paquetes que necesita para dicha orden y prepararlos para su envío. En la OC aparecen las unidades que se tiene que enviar, por lo que el sistema deberá contar cuantos paquetes se deben cargar y de cuanta cantidad cada paquete.

El sistema sugerirá al operario paquetes según el sistema FIFO con las características que precise la OC. El operario podrá elegir cualquier paquete de los que tenga en el almacén, para ello deberá leer con una pistola el código de barras que tiene el paquete asignado. Si las características del paquete no coinciden con lo que está indicado en la OC el sistema deberá mostrar una alerta y no dejar cargar ese paquete. Tras ir seleccionando cada paquete, el sistema irá contando las unidades hasta preparar las unidades indicadas en la OC. El sistema no deberá dejar cargar ni más ni menos unidades.

Tras preparar todos los paquetes con esa información se generará el packing-list. Una vez cargados los paquetes en el camión, se debe realizar un checklist de verificación de envío, esto se realizará desde la aplicación del iAR-Check, pero el envío tendrá asignado el UID del reporte para luego poder hacer referencia a él. Dentro de este checklist, habrá que realizar varias fotos del estado del envío.

Completado el checklist, habrá que generar el albarán, que se utilizará la aplicación que ya está desarrollada o se adaptará. Tras generar el albarán se generará la carta de porte con toda la información del envío, pero el operario deberá rellenar datos del transporte que solo se conocen en ese momento:

- Matrícula del camión

- Datos del transportista
- Fecha
- Firma del transportista (líneas futuras)

Con todo esto se generará la carta de porte del envío. Como paso final habrá que generar un informe del envío para el cliente con la siguiente información:

- Packing-list
- Imágenes del checklist
- Albarán
- Carta de porte

Checklist

Checklist consiste en la web ya creada de iAR. Esta se va a replicar a otra web nueva para el uso propio de la empresa cliente. Como su nombre indica, es un software para la creación, edición y ejecución de checklist digitales. Una forma sencilla y práctica para dejar de lado los formularios y hojas de recogida de datos en papel, así como para eliminar las transcripciones del papel a otros sistemas.

La aplicación web permite la gestión y edición completa de las plantillas de checklist y sus preguntas. Los checklist cumplimentados se pueden consultar y aplicar filtros para su búsqueda, tratamiento y visualización. De cara a la explotación de los datos que se recogen en el proceso (órdenes de fabricación, controles de calidad, rondas, auditorías…), existe un área de visualización a través de gráficas.

En nuestra web, los puntos donde se van a realizar checklists y por lo tanto habrá que asignarle una plantilla del iAR-Check son:

- Recepción de material.
- Checklist periódico de calidad al finalizar la fabricación.
- Checklist de expedición.
- Checklist de parte de trabajo.

Por lo tanto, el usuario primero deberá crear una plantilla de checklist en iAR-Check y luego asignar esas plantillas a los checklist mencionados. Para acceder a los reportes del iAR-Check, también habrá que indicar por parámetro las zonas de las que queremos ver dichos reportes. En la Figura 15 se muestra un pantallazo de la web Checklist de iAR.

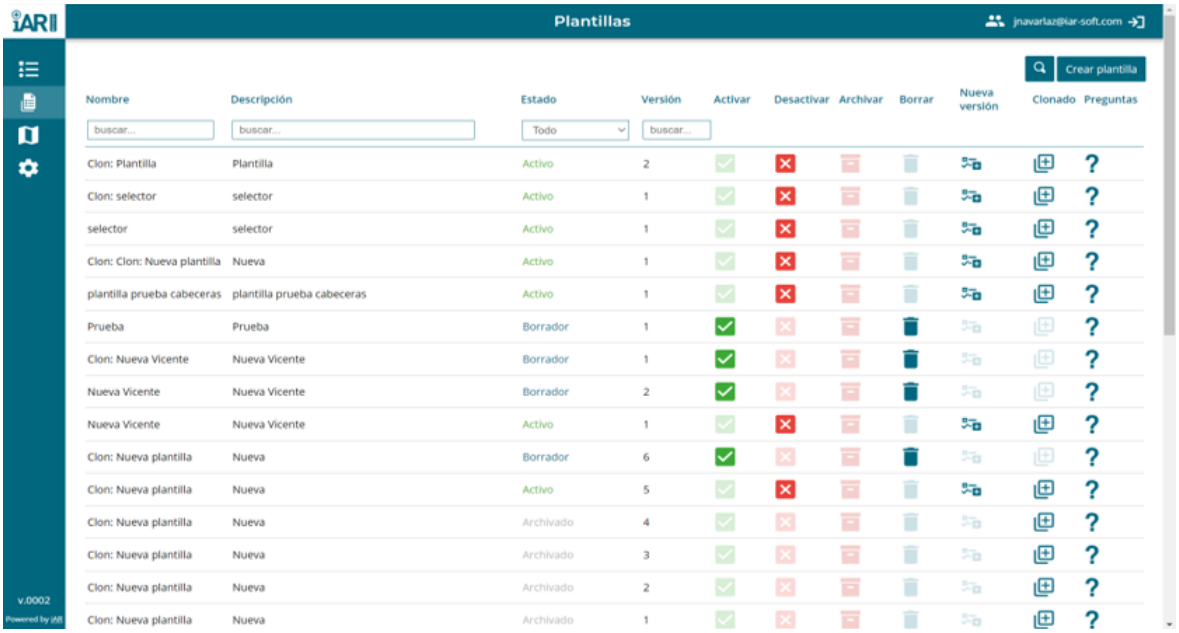


Figura 15. IAR-Checklist

Diseño BBDD

Para la parte de usuarios tenemos el siguiente diagrama de clases con sus respectivas relaciones:

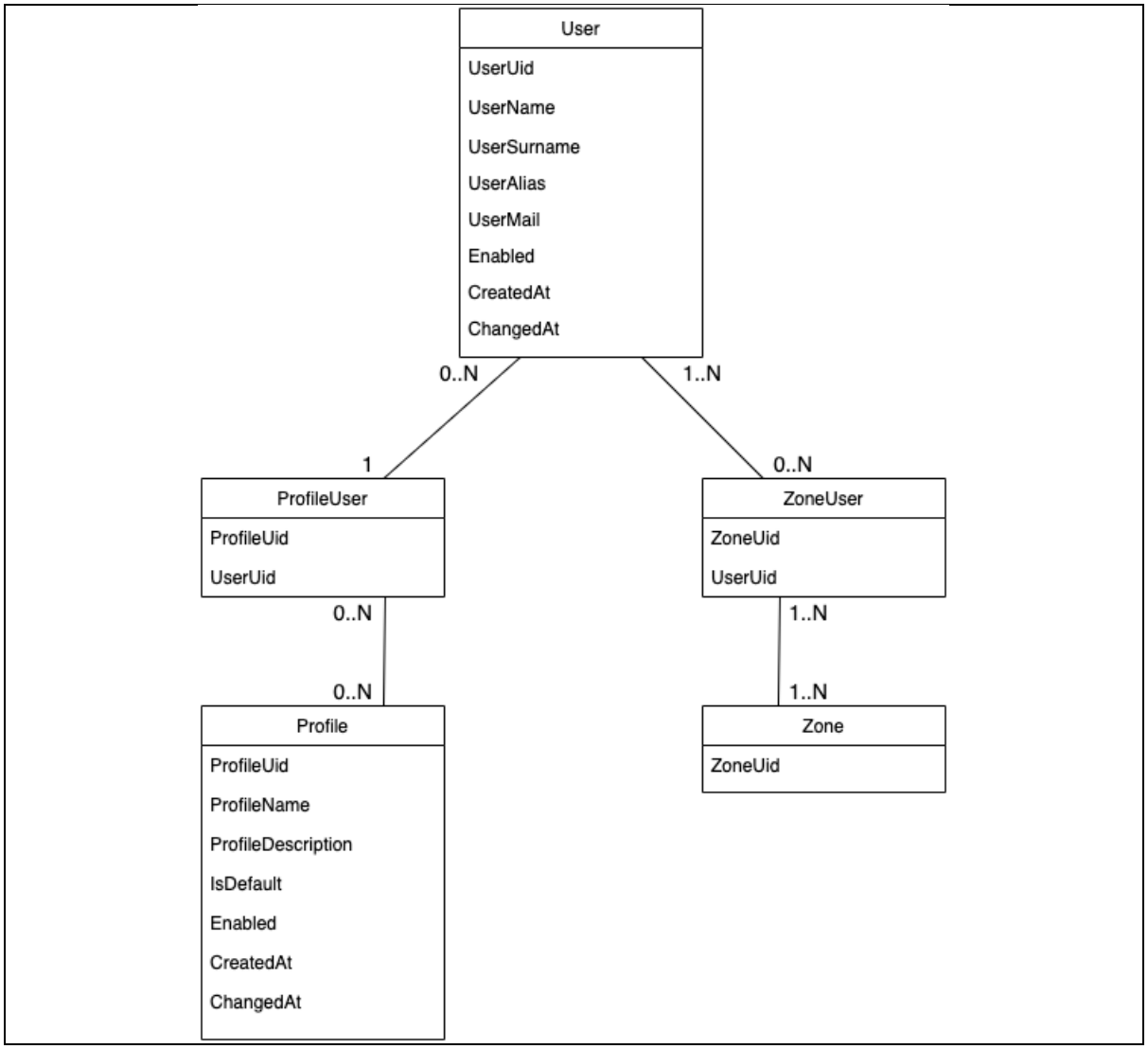


Figura 15. Diagrama Clases Identity

Para la parte de usuarios tenemos el siguiente diagrama de clases con sus respectivas relaciones:

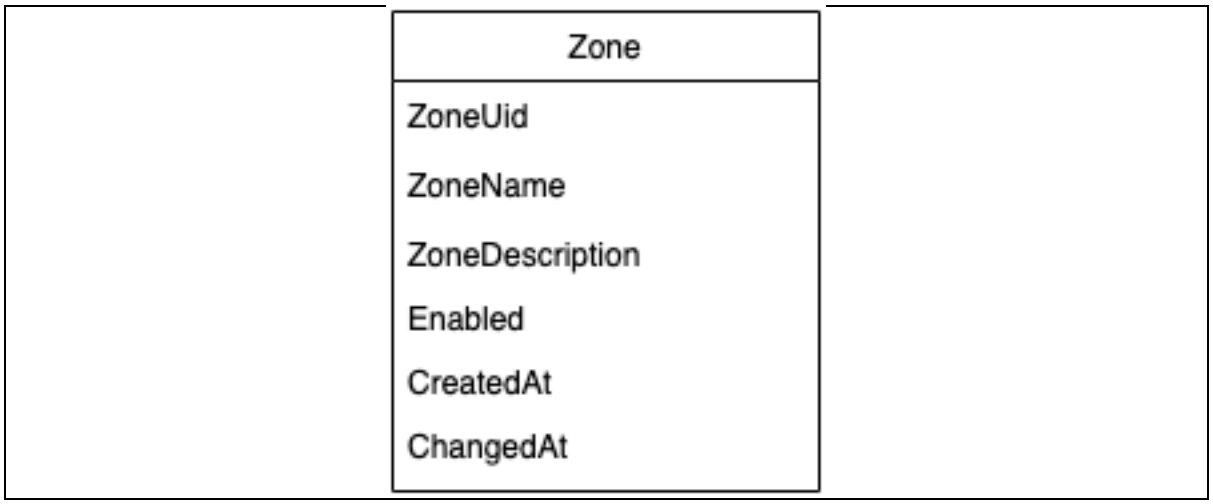


Figura 16. Diagrama Clases Blueprint

Para la parte de las entradas en la fase de producción tenemos el siguiente diagrama de clases con sus respectivas relaciones:

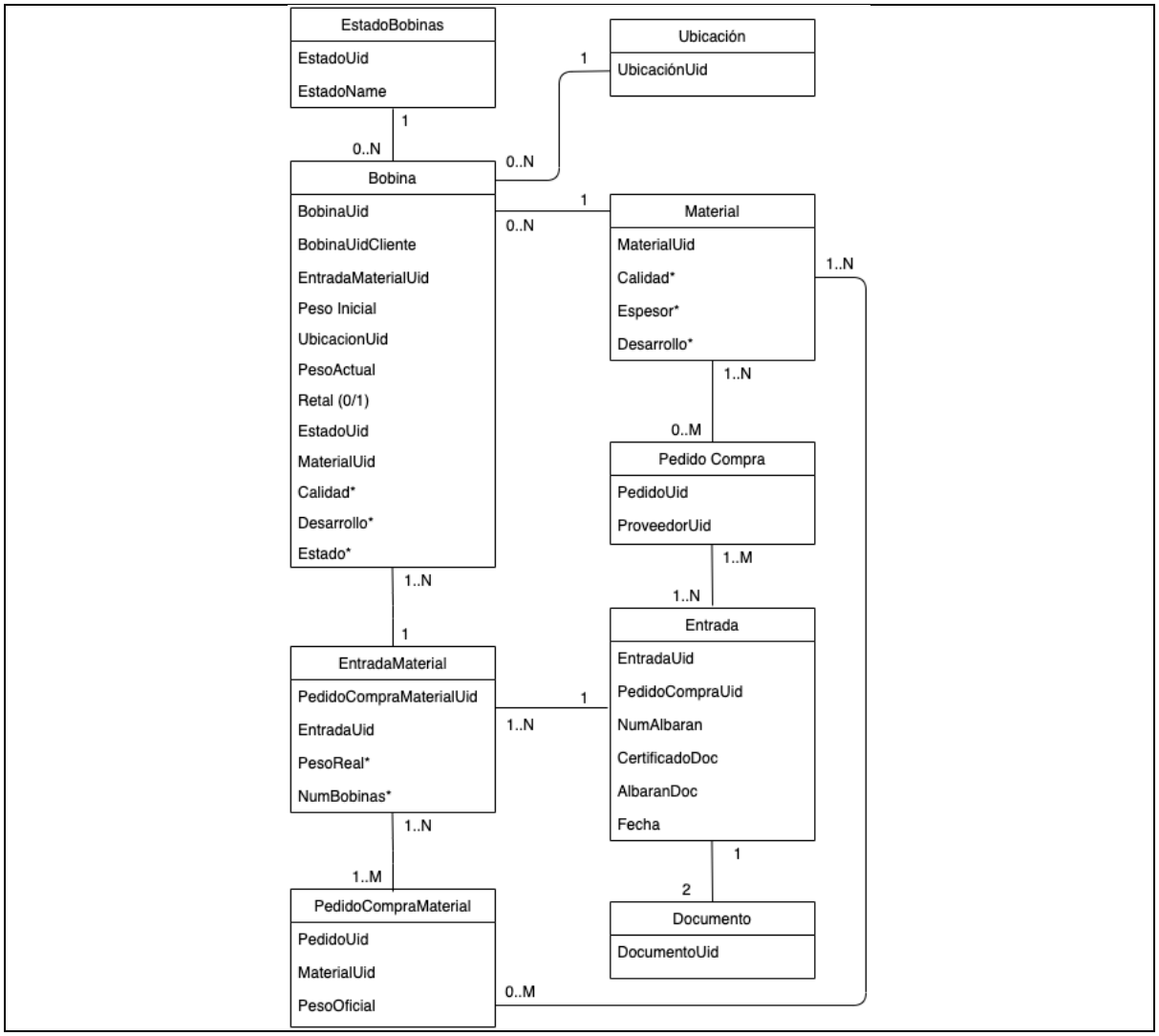


Figura 17. Diagrama Clases Entradas

Para la parte de la fabricación en la fase de producción tenemos el siguiente diagrama de clases con sus respectivas relaciones:

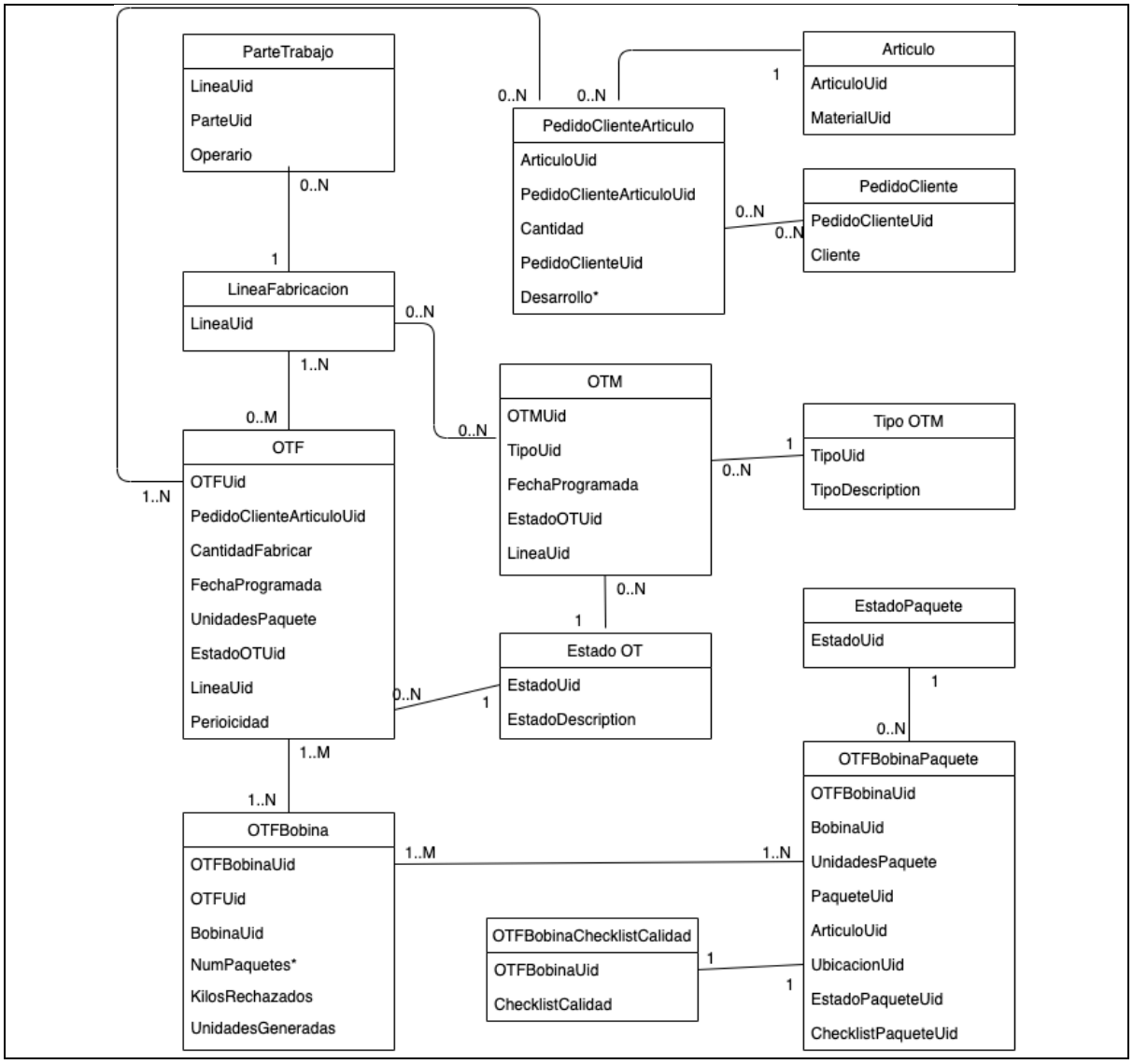


Figura 18. Diagrama Clases Fabricación

Arquitectura

En el proceso de formación de cualquier aplicación, se necesita un entorno de trabajo, donde se pueda desarrollar y modificar un proyecto sin que influya en otras versiones de esta. En el caso de esta web, se cuenta con tres entornos de trabajo, los cuales se utilizarán cada uno para funciones diferentes.

Inicialmente, al principio del desarrollo de la web, hemos trabajado en el llamado entorno de desarrollo, o como usualmente se nombra DEV (Developtment). Este será utilizado para el desarrollo y modificación de toda la web.

iAR cuenta con un servidor general en la nube, que es el que hemos utilizado para todo lo referente al desarrollo. Este cuenta con el servidor de Front end, sevidor de Back end y base de datos. En primer lugar, veremos el servidor de Front o servidor web. Este es un servidor tipo IIS (Internet Information Services), el cual se utiliza para el sistema operativo Microsoft

Windows. Este tipo de servidores ofrece la posibilidad de utilizar en un mismo servidor varios servicios diferentes, es decir, en nuestro caso, tendremos servicios diferentes para cada una de las Web's (Identity, Blueprint,...). En el servidor de Front se encontarán todos los archivos web (HTML, JavaScript, CSS, ...) que compondrán la página Web. Front se encargará de hacer todas las llamadas AJAX necesarias, al servidor de Back. Desde la Web se formarán estas llamadas, y utilizando JavaScript, se lanzarán. Más adelante se explicará cómo se forman y lanzan las llamadas Web Services para obtener la respuesta esperada.

El servidor de Back tendrá los mismos servicios que el de Front. Este recibirá las llamadas de Front, y se encargará de conectarse a la base de datos, y de traducir las Web Services que ha recibido a las consultas SQL necesarias. Una vez haya recibido los datos, los transformará en objetos JSON, que enviará al servidor de Front. La base de datos es SQL, y de cada servicio, tendrá las entidades, relaciones y datos que componen todo el sistema. En esta base de datos del entorno de desarrollo, se han introducido los datos para hacer las pruebas necesarias en la fase de desarrollo. El cliente no nos ha proporcionado ningún tipo de información. En la Figura 19, nos encontramos con el esquema de lo que es la arquitectura del entorno de desarrollo.

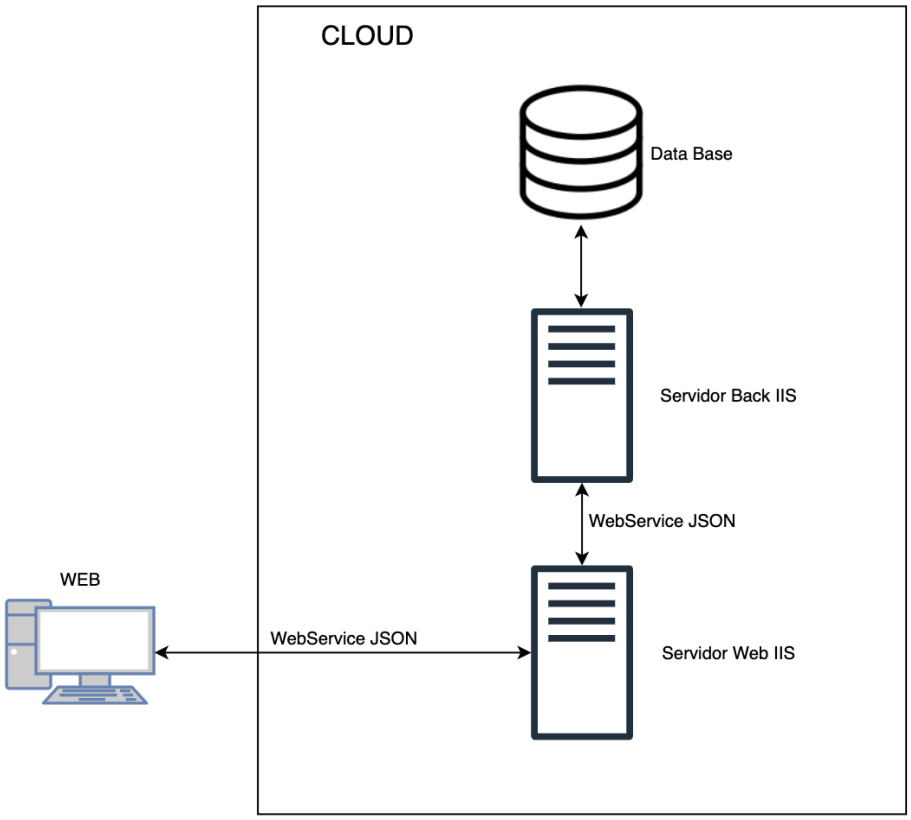


Figura 19. Entorno DEV

Realmente, a lo largo de la jornada, para visualizar los cambios inmediatos y el progreso de la Web en desarrollo, se ha utilizado el servidor web NGINX. Este actúa como proxy de correo electrónico, proxy inverso y balanceador de carga. La estructura del software es asíncrona y controlada por eventos; lo cual permite el procesamiento de muchas solicitudes al mismo tiempo, por lo que, los cambios que se hacían se podían ver inmediatamente en la web. Para utilizar nginx, hay que utilizar en el proyecto una serie de ficheros de configuración, donde se tendrá que especificar cuál va a ser el dominio en el que se encuentra. La IP en la que trabaja este servidor en local es 127.0.0.1, es decir, el localhost. Este es la dirección que apunta el equipo en el que estamos trabajando, desde este mismo equipo. Por lo que, para cada proyecto, el cual estará fuera del mismo, habrá un fichero de configuración, donde se especificará la IP y el nombre DNS de la web que utilizaremos para visualizar los cambios inmediatos. La estructura del fichero se muestra en la Figura 20.

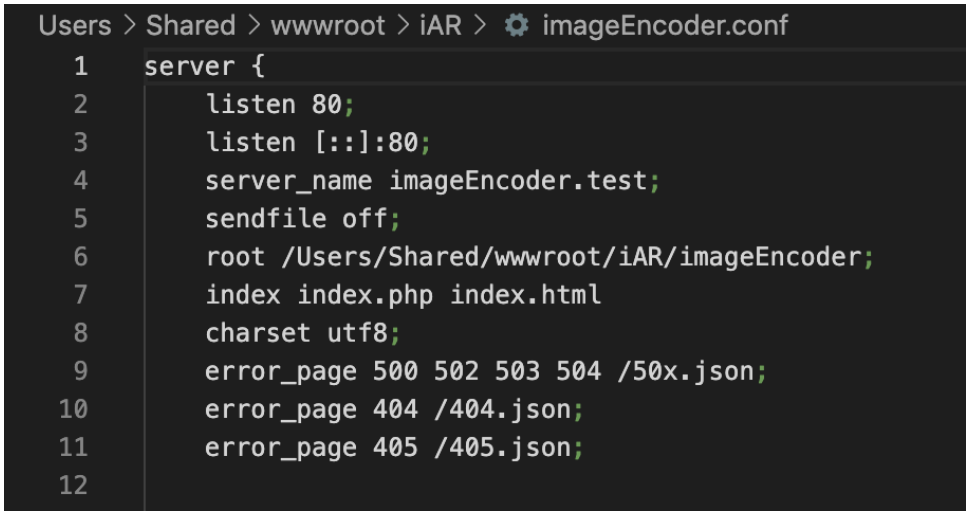


Figura 20. Configuración NGINX

Como vemos, se especifican el puerto, que en nuestro caso es el que está por defecto, el 80, el nombre al que hay que acceder, y la ruta en la que se encuentra nuestro servidor local. Al final de la jornada, subimos todos las modificaciones al servidor en la nube de iAR donde se encuentra el entorno DEV, anteriormente explicado. No es de extrañar que todas las subidas que se hacen al servidor de Dev desde nuestro servidor local, es decir, desde nuestro equipo se hace con Git. Este es un software que generalmente se utiliza para llevar un control exhaustivo en las versiones de un proyecto. Como se ha dicho antes, Visual Studio permite trabajar con Git de manera muy sencilla. Existe la posibilidad de tener los ficheros necesarios dentro de un proyecto, para que, con un simple atajo de teclado, podamos subir los cambios realizados en nuestro servidor local al entorno DEV.

Esta ha sido la forma de trabajar durante la fase de desarrollo. Sin embargo, es evidente que la empresa cliente no va a trabajar con los servidores de iAR, si no que tendrá sus propios entornos de pruebas y producción. Es ahí donde introducimos los entornos PRE y PRO. En el primero, el cliente realizará las pruebas que crea convenientes para verificar todas y cada una de las funcionalidades de la web. El segundo es el utilizado cuando la web se pone en marcha

para su utilización en producción. Estos se encuentran en un servidor local, perteneciente a la empresa cliente.

Este servidor cuenta con una base de datos SQL para PRE (Preproduction), en la cual se encontrarán todos los datos necesarios para las pruebas, una base de datos para PRO (Production), en la cual se encuentran todos los datos reales del cliente. Es sabido también que el cliente utiliza SAP, un software de planificación de recursos empresariales desarrollado por la compañía alemana SAP SE, que incorpora las funciones empresariales claves de una organización. De esta se utiliza una base de datos, donde estarán los que serán creados, modificados y eliminados más frecuentemente. Se utilizará SAP, más concretamente para crear los pedidos de compra y los pedidos de los clientes. Exceptuando eso, el resto de los datos se encontrarán en la base de datos de PRO.

En el servidor de cliente también se encontrarán los correspondientes servidores, tanto de Back como de Front, que tendrán cada uno las mismas funciones que en el entorno de DEV. Una vez en producción, el servidor de Back cogerá los datos de las distintas bases de datos dependiendo de los datos que quiera coger. Para la puesta en marcha de la Web, todo el contenido de los servidores tanto de Back como de Front del entorno de desarrollo se replicarán en los servidores de la empresa cliente. Para ello, se le proporciona a iAR una máquina virtual dentro del servidor local del cliente, donde se subirá todo el contenido tanto de PRE como PRO, configurando las url's de cada uno. Los entornos de Preproducción y Producción se muestran en la Figura 21.

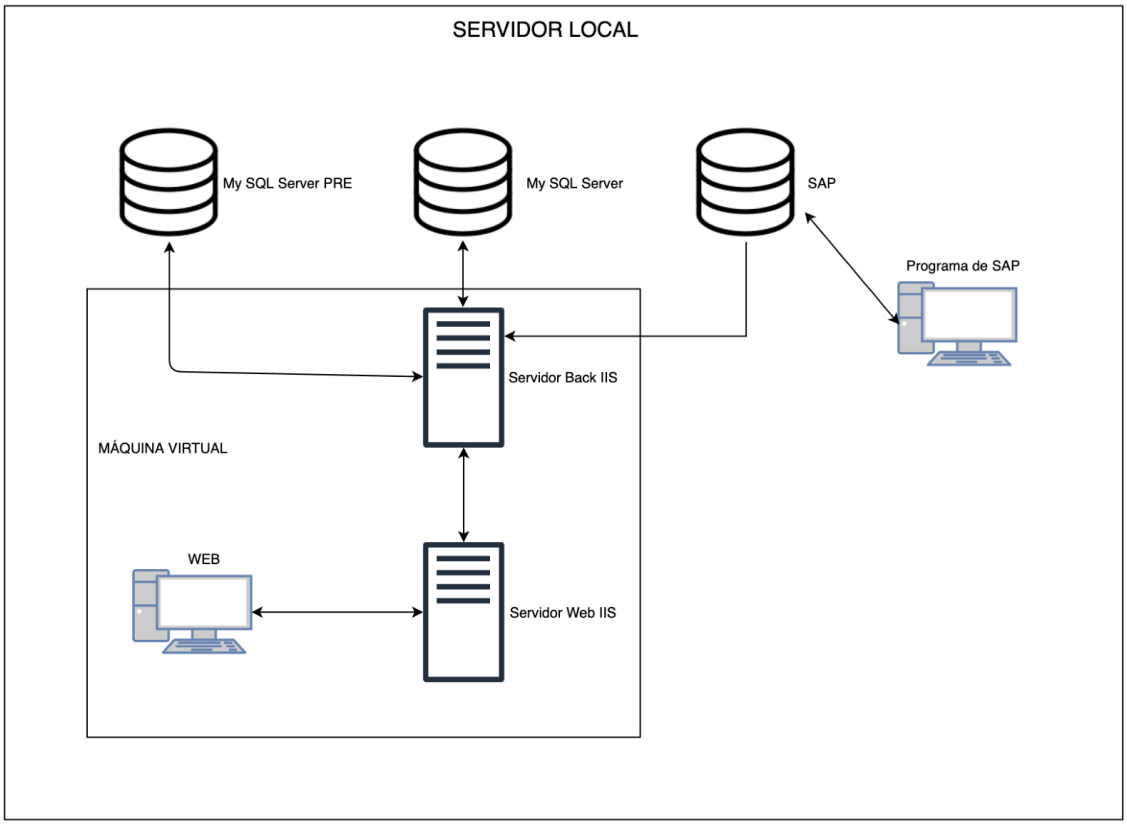


Figura 21. Entrono PRE y PRO

Desarrollo

Visual Studio Code

La herramienta que se ha utilizado para el desarrollo de este producto ha sido Visual Studio Code. iAR utiliza siempre VS para cualquiera de sus proyectos. Esto es debido a que este es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS, que incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. La utilización de Git, la cual va a ser una de las actividades más frecuentes durante todo el desarrollo de la web, se vuelve una tarea muy fácil.

Postman

Postman es una herramienta que se utiliza, sobre todo, para el testing de API REST, aunque también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas. Gracias a esta herramienta, además de testear, consumir y depurar API REST, podremos monitorizarlas, escribir pruebas automatizadas para ellas, documentarlas, mockearlas, simularlas, etc. Una API es el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita del consumidor (la llamada) y el que requiere el productor (la respuesta). Concretamente, la API que vamos a probar es el del proyecto del cliente en el servidor Back de iAR.

Sabiendo esto, para el desarrollo de cada una de las webs, se ha utilizado POSTMAN para probar cada una de las llamadas que se han necesitado hacer, y así poder monitorear las respuestas. De cada uno de los servicios, tendremos los llamados Workspace (sitios de trabajo), donde podremos ver cada uno de los “request” o pedidos que podemos hacer. POSTMAN hace el papel de servicio Web, y realiza llamadas a cada uno de los servicios del servidor de Back. De cada llamada necesitaremos la url, la cual contendrá el servicio al que le queremos consultar los datos, el método (GET, POST, DELETE, UPDATE o PATCH), y los parámetros por los que queremos filtrar el resultado de la llamada. Con estos datos se formará un objeto JSON, que será parte de la llamada.

A continuación, se muestra en la Figura 22 un ejemplo de cómo se forma la llamada, y el formato de la respuesta. Como se puede comprobar, la respuesta es un Array de objetos JSON, las claves de cada objeto coinciden con los campos de la base de datos de una tabla concreta. Pongamos el ejemplo de los usuarios y sus contratos.

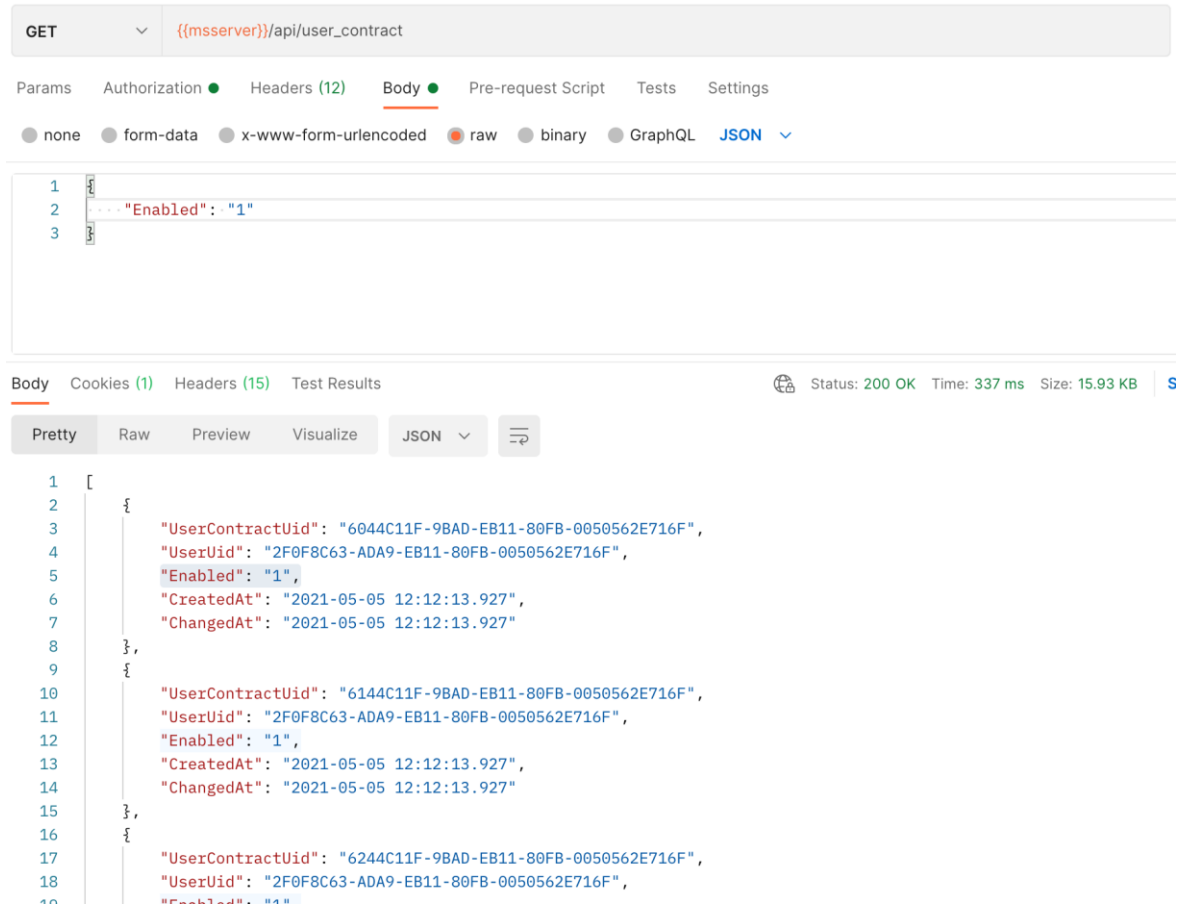


Figura 22. Llamada y respuesta POSTMAN

Nótese que POSTMAN no influye en el funcionamiento de las webs, simplemente se utiliza a la hora del desarrollo. Como ya se ha dicho, de cada servicio habrá un espacio de trabajo, ese espacio de trabajo es un fichero JSON que se ha importado desde POSTMAN.

Con estos datos, se han hecho todas las pruebas necesarias, para probar tanto las peticiones, que se explicarán a continuación, y el correcto funcionamiento de todas las webs.

JavaScript

JQuery (Web Services)

Como ya se ha visto en los puntos anteriores Arquitectura y Postman, el servidor web, es decir, donde se encuentran todos los archivos HTML, JS, CSS, … que componen la web, es la que hace las llamadas REST, las cuales llamaremos Web Services o request (pedido). Para ello se utilizan en JavaScript el método conocido como Ajax (Asynchronous **JavaScript** and XML).

Para cada módulo de nuestras webs, se han definido las Web Services necesarias para realizar el desarrollo deseado, ya sean para crear, modificar o eliminar datos. Para cada uno de los

servicios que nos brinda el servidor de Back end, existen también las request propias del Login y Logout.

Como ya se ha explicado en el punto anterior, para hacer una llamada de este tipo necesitaremos una url. Cada servicio del servidor de Back ofrece distintas urls para hacer distintas llamadas. La primera parte de la url es el nombre del servicio, el cual no se muestra en este trabajo debido a motivos de confidencialidad. Despues, las url varían dependiendo de qué queramos hacer. Para las llamadas también necesitaremos unos parámetros, los cuales servirán de filtro para realizar la Web Service. Finalmente, también necesitamos el método, que indicará el tipo de petición que queremos hacer: GET, POST, DELETE o UPDATE.

iAR posee un componente JavaScript propio que le permite hacer toda la lógica de la llamada con una simple línea de código. Ahora veremos un ejemplo de cómo se forma la request, y como se utiliza el $.ajax() de JQuery. Este última es una librería propia de JavaScript que simplifica la tarea de programar en JavaScript y permite agregar interactividad a un sitio web sin tener conocimientos del lenguaje.

En la siguiente imagen, vemos el ejemplo para obtener todos los usuarios que nos ofrece el servicio Identity que estén operativos.

```
data_web_serv = {
    'url': $CoreIdentity+'/api/user',
    'method': 'GET',
    'parameters': {
        "Enabled": 1
    },
    'id': null,
    'delay': 0
};
list.push(data_web_serv);

sync(list, c_users_list_load_data_return, null, l_users_list_load_data_fail);
```

Figura 23. Web Service Get Users

Como vemos, para formar la Web Service de los usuarios necesitamos una url, en este caso la de Identity y users, el método, que no es más que GET, y los parámetros, que, en este caso, para decir que queremos los usuarios que se encuentran operativos en el sistema en ese momento, tendremos un unico atributo, que debe de ser igual a 1. Utilizando el componente nombrado anteriormente, llamamos a la función sync. A esta función se le pasan un conjunto de parámetros los cuales son:

- El primer parámetro es la lista de request, es un array con todas las llamadas que deseamos hacer.

- El segundo parámetro es el nombre de la función que deseamos que se ejecute si la lista de llamadas se ha realizado correctamente.

- El tercer parámetro son los datos que queremos pasarle a la función anterior en el caso de sea necesario.
- El último parámetro es el nombre de la función que queremos que se ejecute en el caso de que las llamadas hayan fallado.

Ahora se mostrará otro ejemplo, esta vez con el método POST, para ver que los parámetros pueden pasarse tanto por la url como por el método anterior. En este caso se muestra la llamada propia para modificar una zona concreta. Para ello, en la url se encontrará, además del nombre del servicio de Blueprint, el Uid de la zona que queremos alterar. En los parámetros, pondremos los atributos que queremos cambiar de esa zona con los nuevos valores.

```
var data_web_serv = {
    'url': $CoreBluePrint+'/api/zone/'+uid,
    'method': 'UPDATE',
    'parameters': {
        "ZoneName": name,
        "ZoneDescription": description
    },
    'id': null,
    'delay': 0
};
list.push(data_web_serv);

sync(list, c_update_zone_return, null, l_update_zone_fail);
```

Figura 24. Web Service Update Zone

Una vez se ha formado la petición deseada, se utiliza Ajax para mandarlo al servicio correcto en el servidor de Back. Únicamente, de la función sync, se puede mostrar cómo se hace la llamada usando JQuery.

```
return $.ajax({
    url: data.url,
    method: data.method,
    data: data.method=="POST" || data.method == "PATCH" ? JSON.stringify(data.parameters) : data.parameters,
    xhrFields: { //Permite que el cliente envíe las cookies al servidor
        withCredentials:true
    },
    dataType: 'json',
    crossDomain: true,
    beforeSend: function(request) {  //Añadimos las cabeceras necesarias
        request.setRequestHeader("Content-Type", "application/json");
    },
})
```

Figura 25. Función Sync

En el caso de que la llamada se haya dado de manera correcta, se ejecutará la función que le hemos pasado a sync, y en esta función, tendremos como primer parámetro el resultado de la llamada, con la cual podremos hacer lo que queramos en el momento, como puede ser actualizar una lista de usuarios o mostrar la zona recién creada.

Vue

Vue es un framework open source de JavaScript, el cual nos permite construir interfaces de usuarios de una forma muy sencilla. La curva de aprendizaje, desde mi punto de vista, es relativamente baja, claro, debes conocer muy bien JavaScript, saber trabajar con callbacks, promesas, objetos, entre otros temas más [7].

Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes.

En nuestras webs se ha utilizado Vue como componente central, el cual está encargado de llevar la lógica general de toda la web. Vue facilita y simplifica mucho la conexión entre los componentes del modelo vista y controlador. El controlador general es controlado por la llamada instancia Vue. Esta es creada al comienzo de toda aplicación Vue, y está fomada por los datos y métodos. El aspecto de la instancia Vue es el siguiente:

```
$app = new Vue({
    el: '#app',
    router,
    data: {
        signer: {
            signaturePad: null,
            image: []
        },
    },
    methods: {
        load_home: function (){
            $app.$router.push({ path: '/home'});
        },
        load_entries: function (){
            $app.$router.push({ path: '/entries'});
        },
    },
});
```

Figura 26. Instancia Vue

Vue servirá de puente de conexión entre el modelo y la vista. Para ello en el index general del proyecto, que normalmente es .php, accederemos a la instancia Vue, de manera que podremos acceder a ella en todos los módulos del proyecto.

Vue también es perfectamente capaz de impulsar sofisticadas Single-Page Applications cuando se utiliza en combinación con herramientas modernas y librerías de apoyo. Además, ofrece la posibilidad de utilizar estas librerías de modo que puedan transformarse en etiquetas HTML. Actualmente, existen muchas librerías Vue, que son de utilidad para múltiples tareas, sin embargo, existe también la opción de crear componentes Vue propios, de manera que como ya se ha dicho, usarlas como etiquetas HTMLs en las vistas de las webs.

Router

Vue Router es el enrutador oficial de Vue.js (abre una nueva ventana). Se integra profundamente con el núcleo de Vue.js para facilitar la creación de aplicaciones de una sola página con Vue.js [8]. Las características incluyen:

- Mapeo de rutas / vistas anidadas.
- Configuración de enrutador modular basada en componentes.
- Parámetros de ruta, consulta, comodines.
- Ver los efectos de transición impulsados por el sistema de transición de Vue.js.
- Control de navegación detallado.
- Enlaces con clases CSS activas automáticas.
- Modo de historial HTML5 o modo hash, con recuperación automática en IE9.
- Comportamiento de desplazamiento personalizable.

En las webs, utilizamos el router de manera que accedemos a un componente concreto mediante la url. En el controlador general, donde se crea la instancia Vue, se añadirá a la instancia un objeto routes, que es una lista de objetos, el cual cada uno tiene la ruta y el componente al que va asignado esa ruta.

Cada módulo de la web, además de tener modelo, vista y controlador, también tendrá asignado un componente, el cual tendrá métodos propios y datos, los cuales serán proporcionados por el controlador general de la web. Por ejemplo, si la página “Añadir Entrada” tiene modelo, vista y controlador, tendrá también su propio componente, cuyo nombre es el que se indicará en la variable routes de la instancia Vue.

En el index general, hacemos uso de la etiqueta HTML que ofrece el router, la cual tiene el siguiente aspecto:

```
<router-view :datum="current_view"></router-view>
```

Figura 27. Router View

Esta etiqueta tomará el valor del componente que le indiquemos mediante su ruta correspondiente del objeto routes. Por eso decimos que el router de Vue nos permite hacer una web de una sola página. En el controlador, la instancia hace uso de routes para cambiar la vista del componente elegido. Para hacer esto hacemos uso de la función router.beforeEach, donde mediante la instrucción: $app.router.push(path: '/ruta'), cambiaremos el componente del router, y directamente, accederá a la función router.beforeEach, la cual se muestra en la Figura 28.

```
router.beforeEach(function (to, from, next) {
    next();
})
```

Figura 28. BeforeEach

La función next() se encarga de cambiar el componente del index.php en función del path que tenga el router en ese momento.

MVC (Modelo - Vista - Controlador)

Como se ha explicado en la parte de Análisis y Diseño, iAR integra en sus proyectos el patrón de diseño MVC. Este proyecto no es una excepción. También hemos visto en qué consiste cada componente, pero en la práctica, ¿cómo se emplea esto?:

En primer lugar, las vistas son donde se vuelcan y muestran todos los datos que se han recogido de las Web Services.

En segundo lugar, tenemos el modelo, o como se ha llamado en el proyecto caché. Como hemos visto en la explicación de las Web Services, existen funciones donde se configuran las peticiones de las peticiones al servidor de Back. Estas funciones se encuentran en los archivos JavaScript que componen la cache de los componentes de nuestras webs. Por ejemplo, si existe un componente "lista de usuarios", el modelo de este componente tendrá una función llamada lista_usuarios, donde se configurará la primera llamada explicada en el punto de Web Services.

Por último, nos encontramos con el controlador. Anteriormente, se nos ha explicado que a la función sync, propia del componente exclusivo de iAR, se le debe pasar una función que debe ejecutarse cuando cierta petición se ha dado correctamente. Esta función, llamémosla do_ok, se encuentra en el controlador. Como ya hemos visto, el controlador se encarga de toda la

lógica que va detrás de la web. Por lo tanto, del modelo, se devuelven todos los datos al controlador, para que este haga con ellos lo que sea necesario en el momento. Por ejemplo, siguiendo con el ejemplo anterior, una vez el modelo ha recogido los datos de la lista de usuarios, esta es devuelta al controlador, donde este se encarga de volcar todos los datos en la plantilla o la vista del componente, y si fuese necesario, también está encargado de llamar a las distintas funciones del modelo para así realizar una acción u otra.

Para mejor organización, cada uno de los módulos tienen sus componentes separados de los otros. Es decir, en el proyecto no están todas las vistas juntas en una carpeta, ni los modelos ni los controladores. Cada módulo tiene su vista, modelo y controlador juntos, los cuales se comunicarán entre ellos. Por petición especifica de iAR, no se muestran la organización exacta de un componente de las webs.

Componentes - Scanner

Como se ha dicho, Vue ofrece la posibilidad de utilizar componentes propios. Como es el caso, se han creado algunos componentes útiles, como son:

En primer lugar, para crear nuevas entradas, se necesita escanear los códigos de barras del pedido y del albarán. Por tanto, se ha creado un componente Scanner, que dependiendo de los parámetros que se le pasen, podrá escanear códigos de barras o códigos QR.

Este componente constará de un título, un input de tipo texto y un icono de una cámara. Cuando se pulse el icono de la cámara, la cámara del dispositivo se abrirá de manera que pueda escanear el código que enfoque. Tras esto, el valor del código se escribirá en el input.

Para crear un componente Vue, tendremos la instancia de este, y la vista asociada a este componente. En este caso se ha creado el componente v_scanner, que es el nombre que le damos a la etiqueta para utilizarla en la vista de producción correspondiente. Para crear el componente, hay que darle el nombre, el identificador de la vista a la que va asociado, las propiedades que tendrá, que en nuestro caso es un objeto llamado datum, donde estará el parámetro de configuración del tipo de código que queremos que se escanee. También tiene una opción de implementar métodos propios, en los que se realizará toda la lógica del componente, en este caso, el de escanear.

```
scanner_component = Vue.component('v_scanner', {
    template: '#tmpl-scanner',
    name: 'v_scanner',
    props: {
        datum: {
            type: Object,
            default: function () {
                return {};
            }
        },
        allowed: {
            type: Object,
            default: function () {
                return {};
            }
        }
    },
    methods: {
        abrirLector: function() {
            if(!this.datum.show) {
                if(this.allowed.show) { // en el caso de que haya otro componente
                    if(this.datum.code == "cb") {
                        setTimeout(() => {
                            Quagga.init({
                                inputStream: {
                                    name: "Live",
                                    type: "LiveStream",
                                },
                                decoder : {
                                    readers: ["code_39_reader","code_93_reader","code_128_reader","ean_reader","
                                }
                            }, function() {
                                Quagga.start();
                            });
                            Quagga.onDetected(this.onDecodedCB);
                        }, 200);
                    }
                    this.datum.show = true;
                    this.allowed.show = false;
                }
            } else {
                this.datum.show = false;
```

Figura 29. Instancia Componente Scanner

La propiedad es la encargada de almacenar los datos necesarios del escáner, tanto los de configuración como los datos del componente. A la hora de utilizar el escáner, en la vista donde queramos usarlo, meteremos la etiqueta HTML con nombre scanner, y con las propiedades :datum y :allowed le pasaremos los datos necesarios. La manera de llamar al componente es:

```
<v_scanner :allowed="datum.scanners.allowed" :datum="datum.scanners.pedidoCode"></v_scanner>
<v_scanner :allowed="datum.scanners.allowed" :datum="datum.scanners.albaranCode"></v_scanner>
```

Figura 30. Componente Scanner

En el caso de las entradas, le pasaremos los objetos con los datos necesarios, concretamente, el objeto datum debe tener cuatro claves con sus respectivos valores:

- title: título deseado,
- show: true/false,
- info: null,

- code: "qr"/"cb"

Title tendrá el titulo deseado; show indicará si queremos que se muestre la cámara en el componente; una vez decodificado el código que se enfoque en la cámara, su valor se guardará en la variable info; por último, tenemos el parámetro de configuración code, que indicará qué tipo de código queremos decodificar, si código de barras o código QR. La propiedad allowed se utiliza específicamente para que no haya dos componentes de tipo v_scanner abiertos al mismo tiempo, de manera que solo uno este escaneando un código. Por esa razón, se les pasa el mismo objeto a los dos componentes en la Figura 30.

Fijándonos en la Figura 29, en la instancia del escáner, vemos el método abrirLector, el cual es el encargado de abrir la cámara, y dependiendo del tipo de código que queremos escanear, utilizará un tipo de escáner u otro. En el caso de que queramos escanear un código de barras, hemos hecho uso de la librería de Quagga. Esta librería debe su nombre debido a la semejanza que un código de barras tiene a las cebras, cuyo nombre científico es Quagga. Esta librería debe iniciarse, y se tiene que indicar qué tipos de códigos de barras puede escanear. En nuestro caso, por defecto, le permitimos que escanee todos los códigos de barras existentes en el sistema europeo, que son:

- "code_39_reader"
- "code_93_reader"
- "code_128_reader"
- "ean_reader"
- "ean_8_reader"
- "upc_e_reader"

Una vez detecte un código, utilizando la función onDetected propia de Quagga, se le indicará qué método quiere que se ejecute. El método que le hemos indicado es onDecodedCB, que es uo propio del componente scanner.

Para el código QR, utilizamos la librería VueQrcodeReader, la cual abrirá automáticamente la cámara y escaneará el código enfocado. Esta librería está implementada con Vue, por lo tanto, importando la librería, ya podemos utilizar la etiqueta HTML qrcode-stream. Esta es la ventaja de utilizar librerías con Vue, pues le legibilidad del código es mucho mayor.

```
<qrcode-stream
    v-if="datum.code=='qr'"
    class="scanner__read__lector--cam--stream"
    @decode="onDecodedQR">
</qrcode-stream>
```

Figura 31. Componente Scanner

Como podemos comprobar, la diferencia entre un código u otro es abismal a la hora de implementarlo, por eso se dice que Vue simplifica mucho las cosas. La propiedad que le pasamos a este componente es el nombre del método que queremos que se ejecute en el caso de que un código haya sido detectado. En este caso, el método, que será propio del componente scanner tiene por nombre onDecodedQR. En la Figura 32 podemos ver ambos métodos, onDecodedCB y onDecodedQR, donde se cambia el valor del input de la vista del scanner.

```
onDecodedQR: function(decodedString) { //QR
    // console.log(decodedString);
    this.datum.info = decodedString;
},
onDecodedCB: function(decodedData) { //CB
    // console.log(decodedData);
    this.datum.info = decodedData.codeResult.code;
}
```

Figura 32. Métodos del v_scanner

Como hemos visto en el componente, hay un elemento con clave id, que indica a qué plantilla va asignado ese componente. Esta plantilla será la vista del componente, el cual tendrá el id indicado en este. En el caso del scanner, la vista será el archivo PHP con los elementos necesarios. En la siguiente Figura 33 se muestra esta plantilla, donde también se importan las dos librerías necesarias ya nombradas.

```
<template id="tmpl-scanner">
    <div class="section">
        <div class="scanner">
            <div class="scanner__titulo">{{ datum.title }}</div>
            <div class="scanner__read">
                <div class="scanner__read__takeCode">
                    <input class="scanner__read__takeCode__result" v-model="datum.info" type="text" readonly>
                    <img v-if="datum.code=='qr'" class="scanner__read__takeCode__take"
                        src="external_modules/scanner/style/images/qr_scanner_icon.png" @click="abrirLector">
                    <img v-else class="scanner__read__takeCode__take"
                        src="external_modules/scanner/style/images/cb_scanner_icon.png" @click="abrirLector">
                </div>
                <div v-if="datum.show" class="scanner__read__lector">
                    <span class="scanner__read__lector--msg"><?php echo _("Escaneando..."); ?></span>
                    <div class="scanner__read__lector--cam">
                        <qrcode-stream
                            v-if="datum.code=='qr'"
                            class="scanner__read__lector--cam--stream"
                            @decode="onDecodedQR">
                        </qrcode-stream>
                        <div v-else id="interactive" class="viewport"></div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</template>

<script src="/external_modules/scanner/library/js/quagga.min.js"></script>
<script src="/external_modules/scanner/library/js/VueQrcodeReader.umd.min.js"></script>

<script src="/external_modules/scanner/scanner.js"></script>
```

Figura 33. Plantilla de v_scanner

Fijándonos en la vista, podemos ver que se hace uso de todos los datos que se encuentran en el componente utilizando las funciones que nos ofrece Vue, como son:

- v-model: lo utilizamos para el input. Concretamente se utiliza para bindear el valor del input. Cuando cambia nuestro dato info, cambiará automáticamente el valor del input.
- v-if y v-else: si lo que hay dentro de este if es true, el elemento HTML que lo porte aparecerá, en caso contrario no.
- v-on o @: se utiliza para el manejo de eventos. En este caso, por ejemplo, si se pulsa el icono de la cámara, el controlador ejecutará la función abrirLector.

Componentes - Signer

Este componente constará de un espacio propio para realizar firmas. Existe la opción de que en el futuro sea necesario firmar para realizar ciertos checklist. De esta firma, se puede descargar la imagen con la firma, o cifrarla en base 64 para así poder guardar la información y utilizarla en un futuro. Esto se hace pulsando un botón que está debajo del espacio para firmar.

Las librerias que se utilizan en este componente son FileSaver, que se utiliza para descargar la imagen con la firma, y SignaturePad, que se utiliza para habilitar un canvas para que se pueda pintar en él. El componente Signer se muestra en la Figura 34.

```
scanner_component = Vue.component('v_signer', {
    template: '#tmpl-signer',
    name: 'v_signer',
    props: {
        datum: {
            type: Object,
            default: function () {
                return {};
            }
        }
    },
    mounted() {
        var canvas = this.$refs["signature-pad"];
        var signaturePad = new SignaturePad(canvas, {
            dotsize: 6,
        });
        this.datum.signaturePad = signaturePad;
    },
    methods: {
        guardar: function() {
            var canvas = this.$refs["signature-pad"];

            var self = this;
            canvas.toBlob(function(imagen) {

                // Guardar imagen
                saveAs(imagen, "prueba.jpg");

                // Pasar a 64
                var base64 = image_to_base64(imagen);
                base64.then(function(result) {
                    console.log(result);
                    self.datum.image = []; //Comentar esto si quieres mas de una imagen
                    self.datum.image.push(result);
                    //console.log(result);
                });
            });
            this.datum.signaturePad.clear();
        }
    },
    computed: {
        canvasSizeWidth: function() {
            return Math.floor(window.innerWidth*0.5).toString();

        },
```

Figura 33. Instancia Componente Signer

Como se ve en la enterior imagen, la única propiedad que tiene este componente es la de los datos, y a esta se le pasará el objeto con dos valores:

- image: que será un array vacío, donde se guardará un String con la imagen codificada en base 64, en caso de que se quiera guardar en algún momento.
- signaturePad: inicialmente estará a null

Una vez el componente está montado, debemos recoger el elemento HTML canvas que vamos a utilizar como paleta para realizar la firma. Creamos el objeto, SignaturePad con nuestro elemento canvas y lo configuramos según nuestro criterio. En la figura se ve que lo único que cambiamos es el grosor de la línea que es dibujada. El componente está montado cuando toda su estructura ya está creada y todos los datos ya se han asentado. Importante recalcar que no es lo mismo “mounted” que “created”.

Una vez se pulse el botón para guardar la imagen, se ejecutará la función guardar, en la que recogeremos el contenido del canvas y lo transformaremos en una imagen. Después haremos con la imagen lo deseado. Como aún no se ha definido que se hará en un futuro, se ha implementado la posibilidad de descargar la imagen usando la función saveAs que ofrece la librería FileSaver, y también de pasar la imagen a base64. Esto último se consigue con una función propia del componente Signer, llamada image_to_base64, la cual, pasándole un objeto tipo Blob, un tipo de File, te devuelve un String con la codificación en base64 de la imagen. En la siguiente figura se muestra en qué consiste esta función.

```
function image_to_base64(blob) {
    return new Promise(function (resolve) {
        var reader = new FileReader();
        reader.readAsDataURL(blob);
        reader.onloadend = function() {
            var base64data = reader.result;
            resolve(base64data);
        }
    });
}
```

Figura 34. Función image_to_base64

La función devuelve una Promise, un objeto que representa la terminación o el fracaso de una operación asíncrona. Por lo tanto, en la función guardar hay que esperar a que se cargue del todo, lo cual se hace con la función then. Fijándonos también, nos encontramos con la propiedad computada canvasSizeWidth. En Vue, una propiedad computada es la que se actualiza en función del cambio de una característica En este caso, utilizaremos esta propiedad para ajustar el tamaño del canvas en función de la resolución de la pantalla del dispositivo en el que estemos trabajando.

La vista del componente es la que se muestra en la Figura 35, donde están el canvas, y el input tipo botón.

```
<template id="tmpl-signer">
    <div class="section">
        <div class="signer">
            <div class="signer__sign">
                <canvas
                    class="signer__sign--pad"
                    ref="signature-pad"
                    :width="canvasSizeWidth"
                    :height="canvasSizeHeight">
                </canvas>
                <input
                    class="signer__sign--submit"
                    @click="guardar" type="submit"
                    value="Aceptar"
                >
            </div>
        </div>
    </div>
</template>

<script src="/external_modules/signer/library/js/FileSaver.js"></script>
<script src="/external_modules/signer/library/js/FileSaver.min.js"></script>
<script src="/external_modules/signer/library/js/signature_pad.js"></script>
<script src="/external_modules/signer/library/js/signature_pad.min.js"></script>

<script src="/external_modules/signer/signer.js"></script>
```

Figura 35. Plantilla Signer

Las propiedades computadas, como la altura y la anchura del canvas tienen que ser bindeadas utilizando “v-bind:”, o lo que es lo mismo y más simplificado “:”. En caso contrario, estas propiedades no se ajutarán en el momento. La manera de utilizar este componente se muestra en la Figura 36.

```
<v_signer :datum="datum"></v_signer>
```

Figura 36. Componente Signer

Componentes - ImageTaker

Este componente se utiliza para sacar una foto a los certificados en las nuevas entradas. Simplemente consta de un icono de una cámara, en el cual hay que pulsar para que te de la opción de cargar un archivo tipo png o jpg, o de abrir la cámara para sacar una foto y cargarla. De estas imágenes, se guardan sus codificaciones en base 64, para guardarlas.

Tiene dos propiedades: datum, que será la lista de imágenes que se hayan cargado, y config, que es un parámetro booleano, que indicará si el componente puede cargar más de una imagen en el caso de que sea true, o no, en caso contrario. Una vez se cargue la imagen, esta se mostrará en tamaño pequeño. En el caso de que se pueda cargar varias imágenes,

aparecerá la lista de imágenes cargadas, donde cada una tendrá un icono para eliminarla de la lista. El aspecto del componente se muestra en la Figura 37.

```
lector_component = Vue.component('v_image_taker', {
    template: '#tmpl-image-taker',
    name: 'v_image_taker',
    props: {
        datum: {
            type: Object,
            default: function () {
                return {};
            }
        },
        config: {
            type: Object,
            default: function () {
                return {};
            }
        },
    },
    methods: {
        encodeImage: function(e) {
            // this.config.show = false;
            var files = e.target.files;
            console.log(e);
            if((files[0].type).includes("image")) { // Para no subir pdf's o algo parecido,
                var component = this;
                debugger;
                var p1 = read_images_files(files);
                p1.then(function (file_to_save) { // cuando ya se haya cargado p1 del todo
                    var aux = {};
                    aux.file_name = file_to_save[0].file_name;
                    aux.file_ext = file_to_save[0].file_ext;
                    aux.file_mime = file_to_save[0].file_mime;
                    aux.file_binary = file_to_save[0].file_binary;
                    aux.index = component.datum.photos.length;
                    // debugger;
                    if(!component.config.multiple) {
                        component.datum.photos = [];
                    }
                    component.datum.photos.push(aux);
                });
            }
            // this.config.show = true;
        },
        borrarImagen: function(index) {
            this.datum.photos.splice(index, 1);
        },
```

Figura 37. Instancia Componente ImageTaker

En la plantilla existe un input tipo file, donde se cargará una imagen. Una vez tenga una imagen cargada, se ejecutará la función encodeImage, donde se codificará en base 64, como hemos hecho con el componente Signer, e introduciremos el contenido en el objeto datum de ImageTaker. La lista de imágenes se actualizará automáticamente debido al uso que le estamos dando a Vue.

En la Figura 38, se muestra la plantilla de este componente. Con este componente surgió u problema a la hora de desarrollarlo, debido a que una vez se cargaba una imagen, si queríamos cargar una segunda imagen, el input tipo file ya tenía cargado un archivo, por lo que cargaba siempre la primera imagen. Para solucionar esto, se me ocurrió bindear el evento de click. Esto se hace añadiendo al input la propiedad v-on:click, donde siempre que se le

daba click, se ejecutaba la función vaciar, la cual se encarga de eliminar el archivo cargado en el input.

```
<template id="tmpl-image-taker">
    <div class="section">
        <div class="imageTaker">
            <div class="imageTaker__titulo">
                {{ datum.title }}
            </div>
            <div class="imageTaker__takePhoto">
                <div class="imageTaker__takePhoto--take">
                    <label class="imageTaker__takePhoto--take--photo" for="file-input">
                        <img class="imageTaker__takePhoto--take--photo--icon" src="external_modules/imageTaker/style/images/camera_logo.png">
                    </label>
                    <input
                        class="imageTaker__takePhoto--take--input"
                        v-on:change="encodeImage"
                        v-on:click="vaciar"
                        id="file-input"
                        type="file"
                    />
                </div>
                <div class="imageTaker__takePhoto--result">
                    <div v-show="datum.photos.length > 0" v-bind:class="containerClass('photos')">
                        <div v-for="(photo,index) in datum.photos" v-bind:class="containerClass('container')">
                            <img v-bind:class="containerClass('photo')" v-bind:src="binary_data(photo)">
                            <img v-bind:class="containerClass('eliminate')" v-on:click="borrarImagen(index)"
                                src="external_modules/imageTaker/style/images/cancelar_icon_2.png">
                        </div>
                    </div>
                </div>
            </div>

        </div>
    </div>
</template>

<script src="/external_modules/imageTaker/library/js/underscore.min.js"></script>

<script src="/external_modules/imageTaker/imageTaker.js"></script>
```

Figura 38. Plantilla ImageTaker

Para la lista de imágenes, bindeamos las clases, es decir, damos a los elementos HTML Div una clase en función de la longitud que tenga la lista. Esto no es más que para ajustar los estilos de las imágenes en función del número que hay en la lista, concretamente se ajusta su tamaño. En el caso de que haya únicamente una, la imagen aparecerá de mayor tamaño, sin embargo, cuando hay más de una, para que el componente no ocupe mucho la pantalla se reduce el tamaño de cada una de las imágenes.

La forma de utilizar este componente es la siguiente:

```
<v_image_taker :config="datum.certificados.config" :datum="datum.certificados.pedidoCer"></v_image_taker>
<v_image_taker :config="datum.certificados.config" :datum="datum.certificados.albaranCer"></v_image_taker>
```

Figura 39. ComponenteImageTaker

Componentes - TicketPrinter

Este componente complementa al de escaner. Una vez se ha generado la información al escanear un código, el resultado se manda al componente TicketPrinter. Este será el encargado de formar una imagen, ya sea de código de barras o código QR, a partir de esta información. Con esta imagen forma un archivo HTML, y lo pasa vía Bluetooth a una impresora de tickets, la cual imprimirá un ticket con l código deseado. Este componente servirá para poner las etiquetas en los paquetes en la fase de envío en producción.

El componente consiste en una imagen, que puede ser un código de barras o un código QR, y un icono de una impresora, el cual cuando se pulsa se abrirá un selector de dispositivos, para elegir a qué dispositivo deseamos mandarle el archivo HTML. Una vez elegido el dispositivo, que en nuestro caso será la impresora de tickets, con la función sendToPrint se mandará a la impresora.

Tiene una sola propiedad, datum, el cual tiene dos claves: info, que será la información decodificada del código escaneado por scanner, el cual utilizaremos para formar la imagen, y code, que indica qué tipo de código queremos imprimir.

Se utilizan las librerías JsBarcode y vue-barcode, que bindeando su atributo value con la información decodificada, se creará un elemento HTML canvas donde se carga la imagen del código. El funcionamiento es el mismo en ambos componentes. Para la búsqueda de dispositivos bluetooth, se hace uso de la función que tiene el navegador en el que tengamos las webs (se recomienda Chrome), la cual funciona en Chrome, Filezilla o Safari. Con la instrucción navigator.bluetooth.requestDevice() aparece una ventana emergente propia del navegador, con los dispositivos disponibles.

Al igual que en el componente Signer, tomamos el valor del canvas para formar un blob con el HTML que queremos enviar por bluetooth. Una vez tenemos el HTML, hacemos la búsqueda de dispositivos. La búsqueda debe ser configurada, y se tiene que restringir qué tipo de dispositivos deseamos detectar. Por defecto, tendremos el valor acceptAllDevices a true, para que acepte todos los dispositivos que encuentre (esta fución no es la recomendada).

Una vez seleccionado el dispositivo al que queremos conectarnos, debemos encontrar el servicio que este ofrece adecuado para poder mandarle los datos que queremos. Los servicios pueden ser muchos, y estos son valores que pueden ir en hexadecimal, o puede usarse los ids propios de estos, como son:

- BLUETOOTH_CONTROL_SERVICE: 0f291746-0c80-4726-87a7-3c501fd3b4b6
- AUTH_BLUETOOTH_DEVICE_CHARACTERISTIC: ebafb2f0-0e0f-40a2-a84f-e2f098dc13c3
- SHOOTING_STATUS_SERVICE: 8af982b1-f1ff-4d49-83f0-a56db4c431a7
- BATTERY_LEVEL_CHARACTERISTIC: 875fc41d-4980-434c-a653-fd4a4d4410c4
- SHOOTING_CONTROL_SERVICE: 1d0f3602-8dfb-4340-9045-513040dad991
- TAKE_PICTURE_CHARACTERISTIC: fec1805c-8905-4477-b862-ba5e447528a5

Una vez tenemos el servicio que deseamos, necesitamos escribir en la característica propia del servicio para poder mandarlo vía bluetooth. La impresora al recibir el archivo lo imprime, donde está el código que hemos escaneado inicialmente.

Decir que este componente da muchos problemas y aún está en fase de desarrollo. En primer lugar, debe encontrarse la forma de detectar los dispositivos deseados sin la necesidad de aceptar todos, lo cual se recomienda no hacerlo. Después hay que encontrar el servicio necesario y la característica correspondientes. En segundo lugar, da problemas al conectarse con dispositivos Android, como smartphones.

```
printer_component = Vue.component('v_printer', {
    template: '#tmpl-printer',
    name: 'v_printer',
    props: {
        datum: {
            type: Object,
            default: function () {
                return {};
            }
        }
    },
    methods: {
        sendToPrint: function() {
            var canvas = this.$refs["canvas"].$el;
            var self = this;
            canvas.toBlob(function(blob) {
                var p1 = image_to_base64(blob);
                p1.then(function(data) {
                    var image = "<div style='text-align: center'><img style='width: 45rem;' src='"+data+"'>";
                    var info = "<p style='font-size: 2.5rem'>"+self.datum.info+"</p></div>";
                    var oMyBlob = new Blob([image,info], {type : 'text/html'}); // file que hay que mandar
                    navigator.bluetooth.requestDevice(
                        {
                            acceptAllDevices: true,
                        })
                        .then(device => {
                            return device.gatt.connect();
                        })
                        .then(server => {
                            return server.getPrimaryService(0x180F);
                        })
                        .then(services => {
                            return services.getCharacteristic(0x2A19);
                        })
                        .then(characteristic => {
                            return characteristic.readValue();
                        })
                        .then(value => {
                            console.log(value.getUint8(0));
                        })
                        .catch(error => {
                            console.error(error);
                        });
                });
            });
```

Figura 40. Instancia Componente TicketPrinter

En la Figura 41, se muestra la plantilla de este componente. En dicha plantilla podemos ver el uso de las librerías JsBarcode y vue-barcode, donde usando la información, formamos el elemento canvas. A cada uno de los componentes le ponemos la referencia canvas, para recoger su contenido en el controlador.

```
<template id="tmpl-printer">
    <div class="section">
        <div class="printer">
            <div class="printer__icon">
                <vue-barcode
                    v-if="datum.code=='cb'"
                    ref="canvas"
                    class="printer__icon__barcode"
                    :value="datum.info"
                    :options="{ displayValue: false, width: 1, height: 25}">
                </vue-barcode>
                <vue-qrcode
                    v-else
                    ref="canvas"
                    class="printer__icon__qrcode"
                    :value="datum.info"/>
                </vue-qrcode>
                <img class="printer__icon__img" src="external_modules/ticketPrinter/style/i
            </div>
        </div>
    </div>
</template>

<!-- Código de barras -->
<script src="/external_modules/ticketPrinter/library/js/JsBarcode.all.min.js"></script>
<script src="/external_modules/ticketPrinter/library/js/vue-barcode.min.js"></script>
<!-- Código QR -->
<script src="/external_modules/ticketPrinter/library/js/vue-qrcode.min.js"></script>

<script src="/external_modules/ticketPrinter/library/js/FileSaver.js"></script>
<script src="/external_modules/ticketPrinter/library/js/FileSaver.min.js"></script>
```

Figura 41. Plantilla TicketPrinter

La forma de utilizar este componente es la siguiente:

```
<v_printer :datum="datum.printer"></v_printer>
```

Figura 42. Componente TicketPrinter

Para terminar, hay que recalcar que existen los componentes Vue que se utilizarán como pantallas de las webs, como son la pantalla de añadir entrada o gestión de zonas. Entre estos componentes también se encuentran Login o Logout. Estos componentes se mostrarán dependiendo siempre del valor que se le dé al router. Sin embargo, existen de la misma manera aquellos componentes como el header, modal, y los cuatro explicados anteriormente. Estos que serán creados de la misma manera que los otros, se utilizan como si fuesen elementos HTML. Sin embargo, si en el modelo general, en la variable routes, añadiésemos una ruta propia con estos componentes, se comportarán de la misma manera que los componentes que se utilizan como pantallas.

SASS

Para el estilo de la web, evidentemente se ha hecho uso de CSS (Cascading Style Sheets), que significa “Hojas de estilo en cascada”. Este es un lenguaje encargado de estilizar elementos escritos en un lenguaje de marcado como HTML.

Sin embargo, no seremos los desarrolladores los que crearemos los archivos .css encargados de estilizar la web. Para ello se utilizará la herramienta conocida como Sass. Este es un preprocesador de CSS, que nos permite generar, de manera automática, hojas de estilo, añadiéndoles características que no tiene CSS, y que son propias de los lenguajes de programación, como pueden ser variables, funciones, selectores anidados, herencia, etcétera.

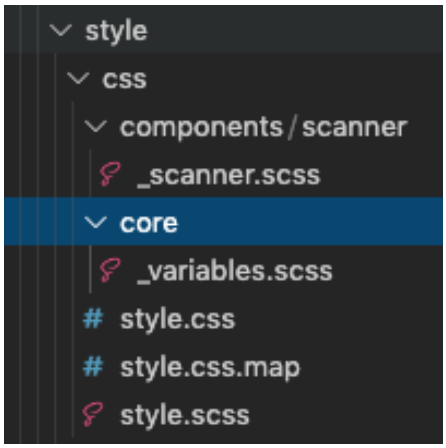


Figura 43. Organización de los directorios de estilos

```
@import 'core/variables';
@import 'components/scanner/scanner';
```

Figura 44. Importaciones de los parciales en style.scss

Una utilidad que ofrece Sass, la cual no soporta CSS, son los mapas. Los mapas son un conjunto de pares <claves>: <valor> que se utilizan para realizar trazas o inspeccionar los estilos de los elementos del DOM llevándonos a los ficheros fuente correspondientes. En nuestro caso, el fichero .map tendrá in elemento cuya clave es “sources”. Su valor es un array, con las rutas de todos los ficheros parciales, de los cuales se hablará más adelante, de quien se escucharán los cambios. El map también tendrá un elemento con clave “file”, y cuyo valor será la ruta del fichero en el cual queremos que se vuelquen los cambios de los archivos que se encuentran en “sources”. Como ya se ha dicho, esto ofrece la modularidad de los estilos.

Por otro lado, Sass soporta todas las reglas @, definidas por CSS3. Además de estas, Sass incluye varias reglas específicas llamadas directivas, de las cuales hablaremos de las que se han dado uso en este proyecto. En primer lugar, la regla @import, la cual ya existía en CSS, sin embargo, Sass la mejora para poder importar también archivos SCSS y Sass. Todos los archivos importados, independientemente de su tipo, acaban fusionándose antes de generar el archivo CSS final. Además, cualquier variable o @mixin definidos en los archivos importados se pueden utilizar en la hoja de estilos principal.

En segundo lugar, se va a dar el caso de que queramos importar un archivo SCSS o Sass pero no necesitamos que se compile como archivo CSS. Para ello, se utiliza un guion bajo como primer carácter del nombre del archivo. De esta manera, Sass no generará un archivo CSS para esa hoja de estilos, pero podrás utilizarla importándola dentro de otra hoja de estilos. Este tipo de archivos que no se compilan se denominan "hojas de estilos parciales" o simplemente "parciales". Aunque el nombre del archivo tenga un guion bajo, no es necesario indicarlo en la regla @import. Así, por ejemplo, si creas un archivo llamado _colors.scss, entonces no se generará un archivo _colors.css. Sin embargo, podrás utilizarlo en tus hojas de estilos con la regla @import "colors";, que importará el archivo _colors.scss. Esto es lo que podemos ver en las figuras Figura 2 y Figura 4, donde se escribe el nombre sin el guion bajo a la hora de importar.

Por último, están las reglas @mixin y @media. Los @mixin se utilizan para producir una salida variada cuando se invocan. Permiten la posibilidad de pasarle parámetros, para así generar la salida correspondiente. Las reglas @media en Sass funcionan prácticamente igual que en CSS, sin embargo, la diferencia está que en Sass es muy fácil definir estilos dependientes de los dispositivos sin tener que repetir los selectores y sin tener que romper el flujo normal de la hoja de estilos Sass. Por tanto, en nuestro caso, nos valdremos de estas dos reglas, ya que la aplicación hará el papel tanto de aplicación de escritorio como aplicación móvil, por lo que tendremos que definir diferentes tamaños de pantalla y utilizarlo para generar estilos según la resolución en la que estemos trabajando.

```scss
@mixin media($size) {
    @if $size == "movile-screen" {
        @media (max-width: 310px) { @content; }
    } @else if $size == "tablet-screen" {
        @media (min-width: 311px) { @content; }
    } @else if $size == "medium-screen" {
        @media (min-width: 711px) { @content; }
    } @else if $size == "big-screen" {
        @media (min-width: 1000px) { @content; }
    }
}

$c_white: white;
$c_grey: rgb(194, 193, 193);
$c_purple: rgb(163,163,243);
$c_azul: rgb(0,78,112);
$c_black: black;
```

Figura 45. Variables creadas mediante Sass

```scss
@import '../../core/variables';

.section {
    .printer {
        &__icon {
            &__barcode {
                width: auto;
            }
            &__qrcode {
                &--icon {
                    color: $c_white;
                }
                width: 70%;
            }
            &__img {
                width: 10%;
            }
        }
    }
}
```

Figura 46. Parcial principal, organizado jerárquicamente

En resumen, Sass se utiliza debido a la claridad que ofrece, a la organización modular de los estilos, lo cual es vital para proyectos grandes, como el que se está desarrollando ahora y

porque proporciona estructuras avanzadas propias de los lenguajes de programación, como variables, listas, funciones y estructuras de control. Además, permite vigilar los ficheros, de tal manera que, si ha habido un cambio en la hoja de estilos, se regenera automáticamente (modo watch). En la Figura 5, se observa la forma en que sass espera a los cambios que se realicen en los .scss o .sass, para volcarlos automáticamente en la hoja de estilos principal "style.css".

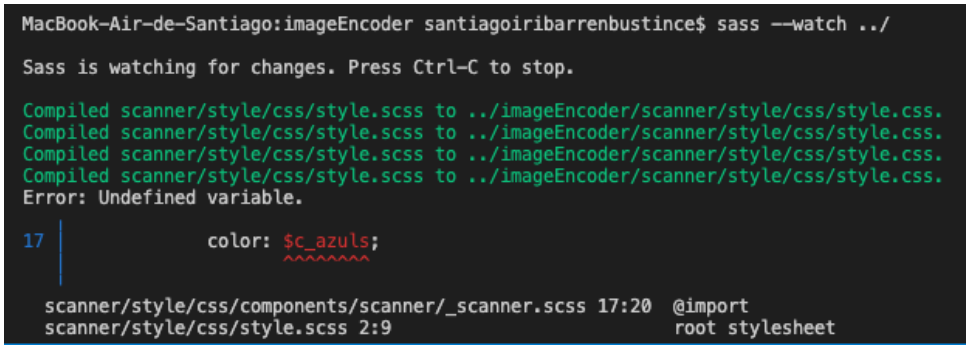


Figura 47. Activación del modo watch

Puesta en marcha

Como se ha visto en el punto de la arquitectura, hemos visto que hay tres entornos distintos de trabajo. Para el desarrollo de las webs, utilizamos el entorno DEV (Developtment). El entorno PRE (Preproduction) y PRO (Production) se encuentran en el servidor local de la empresa cliente.

Una vez se haya acabado todo el proceso de desarrollo, iAR está encargado de utilizar la máquina virtual que le brinda la empresa cliente dentro de su servidor local para subir todo el contenido de las webs a ambos entornos. Utilizando un FTP, configura cada una de las urls para estos dos entornos, de manera que con PRE pueda hacer las pruebas que considere necesarias, y con PRO, que se sube más tarde, una vez se hayan verificado los resultados de las pruebas del cliente, se subirá para ponerlo en producción.

La fase de desarrollo aún no ha acabado, debido a que la web de producción conlleva mucho trabajo, y la fecha límite no ha llegado, pero una vez concluido el proyecto, la subida de dará de la forma explicada.

Conclusiones

El trabajo realizado permite la gestión de los usuarios del sistema, así como de las zonas que forman parte del proceso de producción. También va a permitir utilizar un sistema de gestión integral en la producción de perfiles metálicos, donde se podrá ver y administrar las entradas de materia prima, gestionar todo lo referente al proceso de fabricación y tramitar todas las actividades que esto conlleva. Por último, permitirá dirigir y ejecutar todo lo referente a la expedición de paquetes, resultado de la producción.

Se ha hecho uso del framework de JavaScript Vue, que ha permitido construir una aplicación utilizando el patrón de arquitectura MVC. Con este trabajo se ha demostrado la veracidad de las ventajas que ofrecen tanto Vue como este patrón de arquitectura software, que han permitido un grado elevado de libertad a la hora de organizar el código de los proyectos y la reutilización de distintos componentes.

La aplicación de la metodología ágil para este proyecto también ha sido esencial para el correcto desarrollo de este, donde todos los miembros del equipo de desarrolladores han adquirido aprendizaje en el desarrollo web, y se ha podido dimensionar y organizar mejor el proyecto.

Líneas Futuras

Como ya se ha dicho, entre las líneas futuras del proyecto que podrían tratarse y que no se han implementado por la necesidad de desarrollo en un tiempo limitado, se encuentran:

- Desarrollo completo de la fase de expedición en la web de producción.
    - Desarrollo completo de todas las pantallas y componentes, con sus respectivos modelos, vistas y controladores.
- Introducción en la fase de expedición de la firma de los transportistas (hasta ahora solo introduce la matrícula del camión y los datos de este):
    - Incorporación del componente Vue Signer, donde se guardará el contenido base 64 de la imagen.

Gracias al patrón de diseño que se sigue en las webs, esta permite incluir diferentes módulos que añadirían funcionalidad útil a la web, como son:

- En vez de meter los datos del operario a mano, o los de los transportistas, podrían crearse la clase Operario, donde exista un Uid para cada uno y otros campos que aporten información, como el nombre, y asignarle a cada uno su código de barras respectivo. Para realizar los checklist de fabricación o de expedición, en vez de meter los datos a mano, leer los códigos del trabajador para que los datos se agreguen automáticamente. Nótese que la clase operario no tendría nada que ver con la clase User.
- Es posible en un futuro el cambio del código de barras, al código QR, ya que este último permite codificar un mayor contenido de datos. Este cambio se realizaría alterando la configuración de nuestro componente Scanner. Esta es otra ventaja de la utilización del framework Vue que nos permite hacer en nuestra web.

Bibliografía

[1] "Smart Lean Solutions" https://smartleansolutions.com (accessed May 04, 2021).

[2] "Bambrai Artificial Inteligence" https://bambrai.com (accessed May 04, 2021).

[3] "Industria Navarra" https://www.industrianavarra40.com (accessed May 04, 2021).

[4] "Datos Industria" https://www.navarra.es/NR/rdonlyres/7E1792C2-1BE6-4B9D-993A-CE70FDC011AD/375919/270317de80plan.pdf (accessed May 12, 2021).

[5] "Nexus Integra" https://nexusintegra.io/es/producto/ (accessed May 12, 2021).

[6] "NutSL" https://www.nutsl.com/soluciones/software-de-planta-mes-manufacturing-execution-system/ (accessed May 12, 2021).

[7] "Guía Vue" https://es.vuejs.org/v2/guide/ (accessed May 12, 2021).

[8] "Guía Vue Router" https://router.vuejs.org/ (accessed May 12, 2021).