Wareventory
Una aplicación web para el manejo logístico de almacenes

Universidad Complutense de Madrid
Facultad de Informática
Trabajo de fin de grado
Ingeniería informática
Curso 2023/2024

Autor
Felipe Ye Chen
Director
RAMON GONZALEZ DEL CAMPO RODRIGUEZ BARBERO

RESUMEN

Wareventory, es una aplicación web que permite realizar gestiones y operaciones sobre un inventario logístico de un almacén.

Estos vienen siendo operaciones básicas de cualquier almacén como entrada de trabajos con una cantidad de cajas que se identifican con un PG(Product Group), que contendrá una cantidad de objetos que se identificaran con un código llamado SKU(Stock Keeping Unit). Además de poder realizar un orden de operaciones para el manejo de productos, recibir, ubicar, transferir, recoger y enviar los productos.

El objetivo de este trabajo es un intento de reconstrucción de los sistemas internos que tienen las naves logísticos en concreto 4PX, que comparte una lógica similar con otras empresas de logística y encargadas de cadenas de suministro como pueden ser DHL eCommerce o UPS Supply Chain Solutions de las más famosas.

La implementación de este proyecto está basado en cuanto a memoria y experiencias personales del estudiante, que tuvo la oportunidad de trabajar en uno de los almacenes de la empresa 4PX, por lo que esta aplicación, es un prototipo experimental que busca imitar el funcionamiento básico de los sistemas informáticos utilizados por empresas logísticas. Sin embargo, debe tenerse en cuenta que este desarrollo no es un producto apto para uso real. Su limitada funcionalidad se debe a la falta de colaboración con expertos del sector y a la ausencia de pruebas en un entorno real, lo que impide garantizar su fiabilidad y eficiencia en un contexto operativo.

La implementación final estará hecha de forma local y no hosteado por red, ya que como los sistemas informáticos que se han tomado de referencia, no son accesibles desde red interna, ya que no tiene mucho utilidad el manejar el inventario si no se dispone de los objetos físicos con los cuales se están tratando. además de que puede ser susceptible a ataques en red.

Existen 2 tipos de usuarios dentro de la aplicación:

-Usuario Básico: que dispone de la habilidad de realizar las acciones de manejo de inventario.

-Usuario Administrador: además de poder realizar las funciones del usuario básico, podrá crear y borrar usuarios nuevos, ademas de poder administrar trabajos junto a todos los paquetes relacionados a dicho trabajo

ABSTRACT

Wareventory, is a web application that allows you to perform management and operations on a logistics inventory of a warehouse.

These are basic operations of any warehouse such as job entry with a number of boxes that are identified with a PG (Product Group), which will contain a number of objects that will be identified with a code called SKU (Stock Keeping Unit). In addition to being able to carry out an order of operations for the handling of products, receiving, locating, transferring, collecting and sending the products.

The aim of this work is an attempt to reconstruct the internal systems that logistics companies have, in particular 4PX, which shares a similar logic with other logistics and supply chain companies such as DHL eCommerce or UPS Supply Chain Solutions among the most famous ones.

The implementation of this project is based on the memory and personal experiences of the student, who had the opportunity to work in one of the warehouses of the 4PX company, so this application is an experimental prototype that seeks to mimic the basic operations of the software system used by logistics companies. However, it should be noted that this development is not a product suitable for real use. Its limited functionality is due to the lack of collaboration with industry experts and the absence of testing in a real environment, which makes it impossible to guarantee its reliability and efficiency in a real world context.

The final implementation will be done locally and not hosted over the network, as the software systems that has been taken as a reference, is only accessible from a internal network, since it is not very useful to manage the inventory if the physical objects of which we are dealing is not available, as well as being susceptible to network attacks.

There are 2 types of users within the application:

-Basic User: who has the ability to perform the inventory management actions.

-Administrator user: in addition to being able to perform the functions of the basic user, you can create and delete new users, in addition to being able to manage jobs along with all the packages related to that job.

Indice

1. **Introducción**

En la actualidad, existen muchas aplicaciones similares que sirven para el manejo de inventario, ya sea en tiendas online o como en almacenes logísticos donde no se encargan de vender los objetos sino de almacenarlos y enviarlos cuando son necesarios.

Esta es una aplicación más que se une al grupo, pero en este caso, se intenta aportar una visión nueva en cuanto al diseño del código y a la flexibilidad y escalabilidad.

En este documento se desarrollará la diferencia de esta aplicación, herramientas y tecnologías utilizadas, las funcionalidades que incluye junto a la arquitectura del proyecto y las conclusiones del desarrollo de esta aplicación.

Palabras claves:

SKU(stock keeping unit): es un código que se asigna a un producto para identificarlo y rastrearlo en inventarios y sistemas de gestión.

PG(package group): es un código que se asignan a las cajas contenedoras de productos para identificarlos y registrarlos en el sistema.

1.1 Motivación

Dado al entorno de crecimiento que he tenido, he podido ver cómo se lleva el negocio en tiendas sin sistemas informáticos(tiendas de ropas, bazares, etc), otros con sistemas básicos y por último un sistema informático completo en las naves logísticas. Por lo que he tenido una buena cantidad de experiencia en cuanto cómo organizan el inventario distintos grupos de personas.

La propósito e idea principal de esta aplicación web, es la fragmentación, con la arquitectura y tecnologías usadas se ha conseguido fragmentar todos los componentes usando la arquitectura hexagonal, teniendo controlado todos los datos con los que se trata además de conservar las ventajas de la arquitectura hexagonal de hacer el código más escalable y mantenible.

Un aspecto adicional de este diseño es para que una persona con un conocimiento básico de programación, siguiendo una documentación pueda modificar el código a su gusto y poder implementar más funcionalidades según a su gusto y sus necesidades, en vez de ser una aplicación inmutable con todo predefinido, dado que este proyecto se encontrará de forma open-source en github.

1.2 Planificación

Para realizar los objetivos de este proyecto se ha dividido el trabajo en varias fases de desarrollo que se resumen en las siguientes fases:

-Diseño y creación de la estructura de datos y base de datos
-Gestion de usuarios y verificacion web con JWT token
-CRUD de datos
-Documentacion

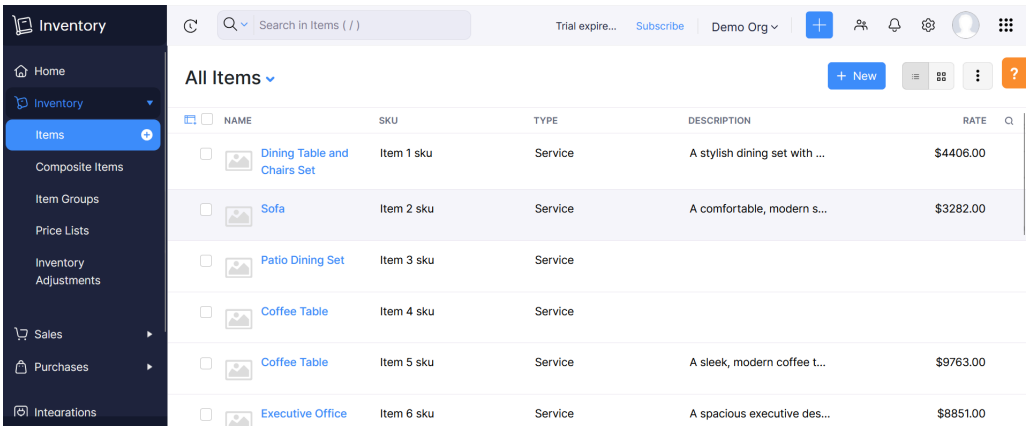
2. Estado de la situación

En este apartado se hará un análisis de las diferentes herramientas ya existentes en el mundo para el manejo de inventario y se mostrará la diferencia que tiene nuestro proyecto sobre el resto.

2.1 Aplicaciones similares

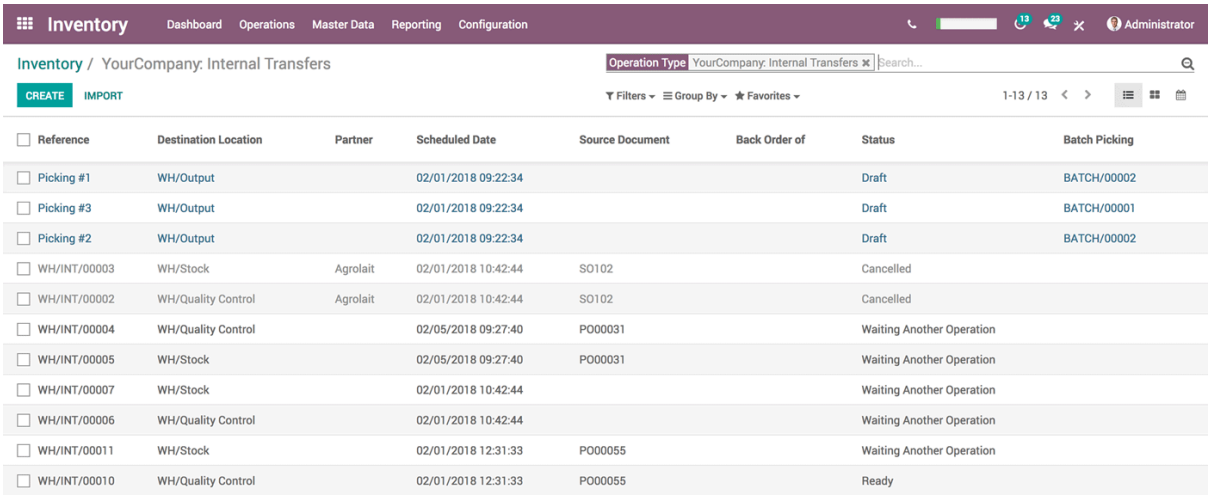
2.1.1 Zoho Inventory

Zoho Inventory es una aplicación de gestión de inventario diseñada para pequeñas y medianas empresas. Permite una gestión del inventario de forma intuitiva además de la gestión de órdenes de venta y compra.



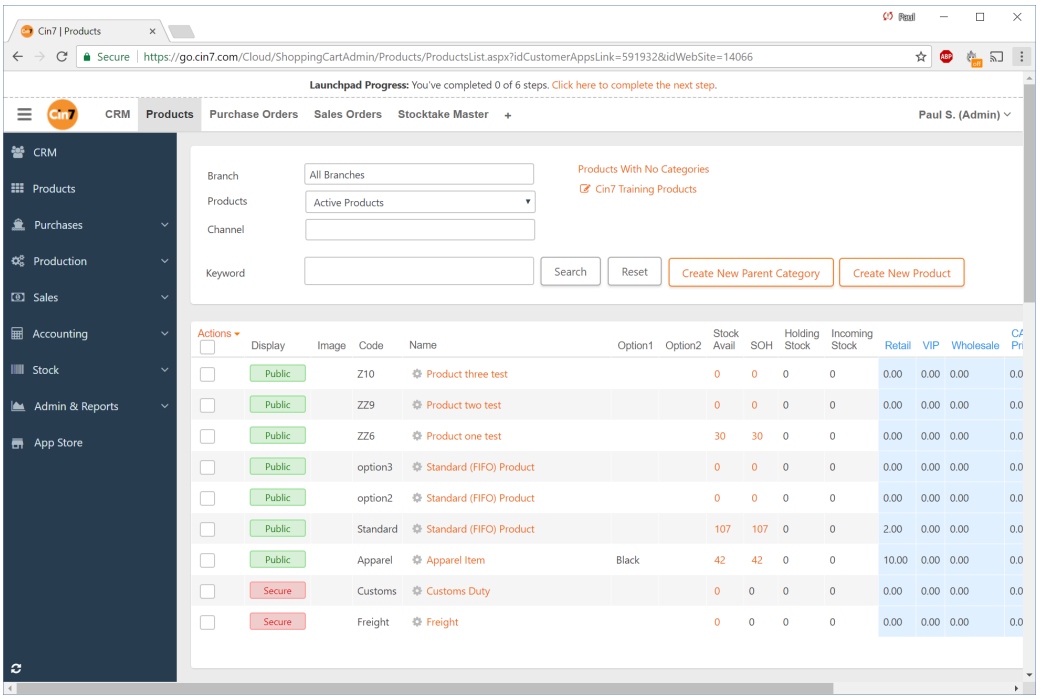
2.1.2 Odoo

Odoo es la aplicación más popular en el momento de ERP(enterprise resource planning) que aparte de proporcionar un manejo de inventario, incluye muchas más operaciones como pueden ser contabilidad, ventas, compras, CRM, y recursos humanos, entre otros. Además de que es una aplicación Open-Source con muchas modalidades y opciones haciéndolo escalable para todo tipo de empresas.



2.1.3 Cin7

Cin7 es una aplicación que permite al igual que los ejemplos anteriores un sistema de manejo de inventario, con ventas, automatización de pedidos y un sistema de contabilidad, y plataformas de e-commerce. Esta es una aplicación que requiere el pago de una suscripción para su uso



2.2 Valores y diferencias de nuestra aplicación

Nuestra aplicación reúne varias funcionalidades centradas solo en el manejo de inventario, ya que es el área con el se es familiar con. Y la idea tampoco es replicar odoo que es Open-Source, sino hacer una aplicación con ideas de arquitectura diferentes.

Una diferencia principal de este proyecto es el énfasis en tener una ubicación constante de todas las ubicaciones, que aplicaciones como Zoho no tiene implementado y no se puede añadir dado que el código fuente no es público.

Otra diferencia notable es que la idea que se presenta esta aplicación es menos restrictiva, en las dos operaciones más relevantes se da una opción siguiendo el protocolo y otra donde se realiza una operación directa, el protocolo en almacenes como 4PX del manejo de inventario es el siguiente:

Cuando llega una caja, se crea una lista del trabajo, se recibe escaneando el código de la caja y los códigos de los objetos, se ubica con el código y la ubicación,

para enviarlos se genera una lista de cada pedido y se recoge siguiendo la lista y por último se empaqueta.

En cuanto a la forma directa, se ha hecho para que se pueda recibir directamente un objeto sin necesidad de un trabajo nuevo, simplemente introduciendo un código identificativo y la cantidad, luego se ha de ubicar en una localización y se puede enviar directamente introduciendo el código, origen y cantidad.

Otro valor añadido es la fragmentación de todas las componentes y una separación entre BackEnd y FrontEnd, esto permite un mejor entendimiento de código en cuanto a nivel de programador, que permite realizar pruebas unitarias sobre cada elemento sin dependencias de otros factores.

3. Tecnologias, arquitecturas y conceptos aplicadas

En este apartado se procederá a explicar las tecnologías y diferentes decisiones que se han tomado a la hora de desarrollar la aplicación web, que como todos, se dividirá en dos partes, FrontEnd y BackEnd.

3.1 BackEnd

Este proyecto ha empleado python-flask como framework de desarrollo que realizará las operaciones del lado de servidor, que se encargará de devolver las respuestas que correspondan y conectar con la base de datos, la cual está implementada usando MongoDB como base de datos para el almacenamiento de la información.

3.1.1 Python-Flask

Es un framework de python diseñado para ser ligero y flexible, conocido por su simplicidad y arquitectura minimalista, que no dispone de una estructura rígida como puede ser Django que está más orientado a base de datos SQL.

Se ha escogido este framework dado que es fácil de usar y de aprender, además de que los paquetes de python son fáciles de instalar con pip install, amplios y están bien documentados

3.1.2 MongoDB

MongoDB es una base de datos noSQL, que almacena los datos en formato de JSON, que permite una estructura más dinámico sin esquemas rígidos, en este proyecto que se centra en el manejo de inventario, se tienen muchos datos que son arrays y de tamaño variable, por eso se ha elegido este tipo de base de datos noSQL para el almacenamiento de información.

3.1.3 MongoEngine

Aqui se incluye MongoEngine y PyMongo, ambos son paquetes instalados con pip install, estas librerías permite una facil conexion con la base de datos de MongoDB, que además incluye clases definidas como son StringField, IntField, que sirven para definir el tipo de datos que representa una variable, Document , EmbeddedDocument y DynamicDocument que sirven para definir objetos, subobjetos y objetos dinámicos.

3.1.4 flask_CORS

Dado que el FrontEnd y el BackEnd están separados, al hacer la conexión desde Front a Back el error más común es que este bloqueado el acceso, dado a la política por defecto de negación de CORS (**Cross-Origin Resource Sharing**), se ha de usar librerías como flask_CORS para poder establecer conexiones desde otro origen.

3.1.4 JWT (JSON Web Token)

Esta librería se usa para crear tokens de identificación, donde guardaremos de forma cifrada con una clave secreta datos como el nombre de usuario, su rol y el propio token. Esto nos permitirá crear funciones adicionales que comprueben el token en los headers de las peticiones http y determinar si el usuario puede acceder a ciertas rutas o no.

3.2 FrontEnd

El FrontEnd de este proyecto está hecho con Angular y una plantilla gratis que proporciona PrimeNG, que agiliza la construcción visual de componentes.

3.2.1 Angular

Angular es un framework de desarrollo web de código abierto creado y mantenido por Google, diseñado para construir aplicaciones web de una sola página (SPA, por sus siglas en inglés) de manera eficiente y escalable. Angular utiliza TypeScript, un superconjunto de JavaScript, para ofrecer un entorno de desarrollo robusto y estructurado que facilita la creación de aplicaciones complejas y dinámicas.

Este framework sigue una arquitectura basada en componentes, lo que permite a los desarrolladores dividir la aplicación en partes modulares, reutilizables y fáciles de mantener. Angular también ofrece una serie de herramientas y características integradas, como el enrutamiento, la gestión de formularios, la inyección de dependencias, y un sistema de compilación avanzada, que optimizan tanto el desarrollo como el rendimiento de las aplicaciones

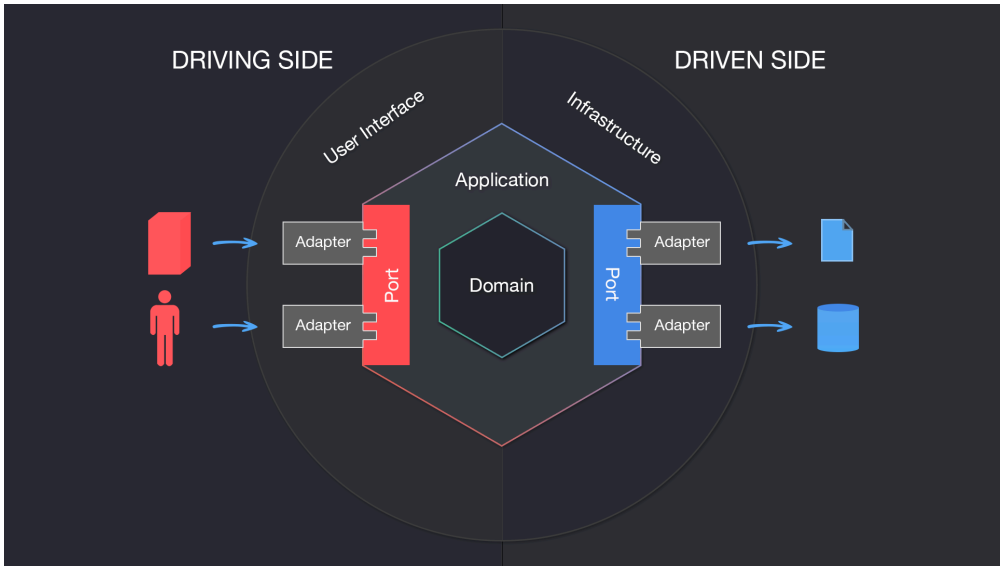
3.2.2 Arquitectura Hexagonal

La arquitectura hexagonal, también conocida como arquitectura de puertos y adaptadores, es un enfoque de diseño de software que promueve la separación de las responsabilidades y la independencia entre los componentes de una aplicación.

Esta arquitectura se estructura en torno a un núcleo central que contiene la lógica de negocio (el dominio), rodeado por capas externas que interactúan con el mundo exterior a través de puertos (interfaces) y adaptadores (implementaciones).

El objetivo principal de la arquitectura hexagonal es hacer que el núcleo de la aplicación sea independiente de los detalles técnicos, como bases de datos, interfaces de usuario o servicios externos. Esto se logra permitiendo que los componentes externos se conecten al núcleo a través de puertos bien definidos, mientras que los adaptadores implementan las conexiones necesarias para interactuar con tecnologías específicas. De esta manera, es posible cambiar, agregar o sustituir componentes externos sin afectar la lógica de negocio, lo que mejora la mantenibilidad, escalabilidad y flexibilidad del software.

En nuestro caso los puertos se encuentran en la carpeta de “application”, el dominio en una carpeta de “domain” y la implementación en “infrastructure”, esto se explicará más detalladamente en apartados posteriores.



3.2.3 PrimeNG

PrimeNG es una biblioteca de componentes de interfaz de usuario (UI) de código abierto para Angular, desarrollada por PrimeTek. Ofrece una amplia colección de componentes ricos y personalizables que facilitan la creación de aplicaciones web modernas y con un diseño atractivo.

Uno de los principales atractivos de PrimeNG es su enfoque en la productividad y la facilidad de uso, proporcionando una experiencia de desarrollo fluida y consistente. Además, cuenta con temas predefinidos y herramientas de personalización que permiten a los desarrolladores adaptar la apariencia de los componentes a las necesidades específicas de sus proyectos.

3.2.3 Sakai Template

Sakai Templete es una plantilla gratis proporcionado por PrimeNG que dispone de componentes más sofisticados y con una hoja de CSS global, que mantiene una uniformidad visual a través de dicha plantilla

3.3 Tecnologías generales

3.3.1 MongoDB Compass

MongoDB Compass es una interfaz gráfica de usuario (GUI) desarrollada por MongoDB, diseñada para facilitar la gestión y análisis de bases de datos MongoDB. Proporciona una forma intuitiva y visual de interactuar con las bases de datos

3.3.2 Github

GitHub es una plataforma de desarrollo colaborativo que utiliza el sistema de control de versiones Git para gestionar y almacenar código fuente en la nube, que facilita la colaboración entre desarrolladores mediante la creación de repositorios de código, la gestión de versiones, y la implementación de pull requests para revisar y fusionar cambios. Ofrece herramientas para la integración continua, la documentación del proyecto, y la gestión de tareas, promoviendo un flujo de trabajo ágil y eficiente.

3.3.3 Visual Studio Code

Entorno de desarrollo y editor de código desarrollada por Microsoft, siendo ligero y customizable con gran variedad de plugins.

3.3.4 Copilot

Siendo 2024 se ha usado la ayuda de la IA para poder solucionar problemas y ver soluciones de código de forma más rápida, el uso de esta herramienta ha sido de gran ayuda en cuanto a encontrar errores y para aprender sobre conceptos nuevos.

3.3.5 i18n

i18n es una abreviatura comúnmente utilizada para referirse a "internationalization" (internacionalización) en el desarrollo de software. Que en este proyecto se ha usado para la creación de una estructura que permite la traducción de textos y mensajes,

4. Arquitectura de proyecto

4.1 Despliegue de la aplicación y desarrollo local

Para poder desplegar la aplicación web de forma local, se necesita de los siguientes requisitos iniciales:

-Python 3.X versión de python superior a 3.0
-Node.js 17.X versión de NodeJs alrededor de 17.0 o superior
-Servidor Mongo con MongoDB compass

Para la instalación de dependencias de BackEnd es recomendable usar un entorno virtual para realizar la instalación de paquetes de pip, si se obtiene el código desde el github, se ha de seguir los siguientes pasos

- cd BackEnd
- python venv .env
- .env/Scripts/activate
- pip install -r ./requierments.txt
- py db_scripts.py
- py main.py

Con las instrucciones anteriores se debería de haber generado una base de datos con datos de prueba y empezado el servidor backend en http://127.0.0.1:5000 o http://localhost:5000

Para el arranque de la parte FrontEnd se ha de seguir los siguientes pasos:

- cd FrontEnd
- npm install -g @angular/cli
- npm install
- ng serve

Con esto debería tener todo lo necesario para poder desplegar la aplicación de forma local y tener todas las funcionalidades disponible

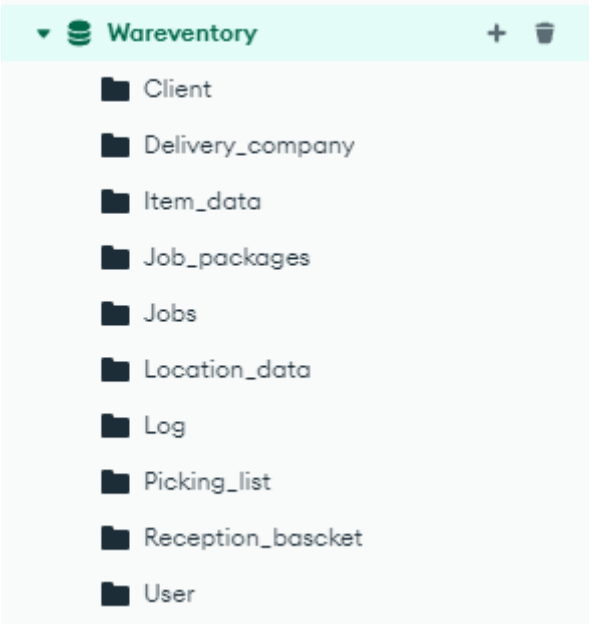
4.2 Generación de Base de datos

Para realizar pruebas iniciales con datos se ha creado un script en python que genera valores aleatorios en un archivo ubicado en /BackEnd/db_script.py

La base de datos por defecto se encuentra en mongodb://localhost:27017 y el nombre de la colección es Wareventory y se tiene varias cuentas de prueba, todas con una contraseña por defecto de “123456” que se puede cambiar en la linea 30, para el administrador el usuario es “ESJ75089” y varios usuarios básicos generados aleatoriamente y uno fijo con nombre de usuario “aaa”.

4.2.1 Estructura Base de Datos

En este apartado se va a proceder a la explicación de las diferentes colecciones que existen dentro de nuestra base de datos



- Client: nos servirá para guardar la información de las personas que guardan objetos en nuestro inventario, que contiene un código de cliente, nombre, número de teléfono, email y dirección.

- Delivery_company: esta guarda información sobre las empresas de reparto que se usará, contiene al igual que el cliente, un código, nombre, número de teléfono, email y dirección.

- -Item_data:guarda información de los objetos, contiene un SKU, un array que incluye todas las ubicaciones que se encuentra el objeto,

una cadena string para almacenamiento de imágenes, sus dimensiones y su peso.

- Jobs: contiene información general sobre un trabajo, que tiene un código de trabajo, el suministrador, que es uno de los clientes, el tamaño del trabajo, es decir, cuantas cajas contiene dicho trabajo,método de llegada, una descripción breve y si esta cerrado, esto sirve si se quiere dejar de trabajar sobre un trabajo ya que puede existir algún error con la comunicación y se pueda poner el pause el trabajo

- Job Packages: almacena información de cada caja registrada en los trabajos, contiene un código de trabajo, un código PG, que es el código de la caja, el SKU del objeto que contiene dentro de la caja, la cantidad esperada, recibida y ubicada.

- Location_data: contiene el código de ubicación, y una array item_data que solo incluye el SKU y la cantidad disponible.

- Log: aquí se almacena un historial de todas las acciones que se han hecho, cualquier recepción,ubicación, etc. Esta es una colección dinámica, es decir que almacena datos diferentes dependiendo de la acción registrada. Si se registra un historial de transferencia entre ubicaciones se almacenará siempre el usuario de la accion, la accion en si y el tiempo actual, después se almacena un origen y un destino, el SKU del objeto y la cantidad, mientras que si se registra una acción de recepción, solo se almacenará el código del paquete, el SKU y la cantidad.

- Picking_list: aquí se almacena un código de lista, el array de objetos que se deben de recoger y el estado de la lista, que puede ser “Pending”, “In Progress”, “Completed”

- Reception Basket: contiene una lista de los objetos que se han recibido, con su SKU, cantidad, código de trabajo y código de caja, esta colección sirve para tener almacenados los objetos recibidos y se saquen de esta lista cuando se quieran ubicar dichos objetos

- User: almacena los datos de usuarios, el código de usuario, contraseña almacenado con hash, un rol que puede ser “admin” o “basic”, un DNI y el nombre.

4.3 Arquitectura FrontEnd

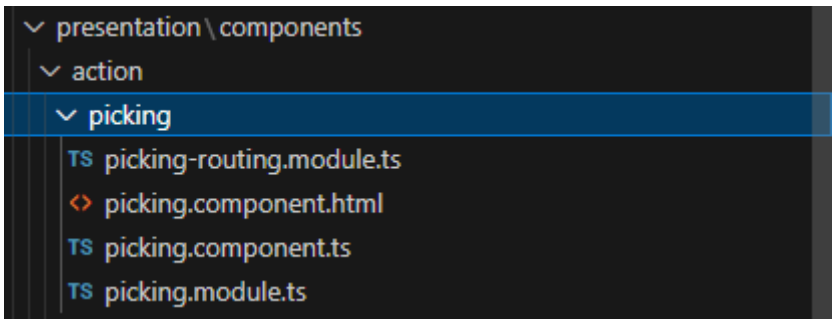
Como se ha explicado en apartados anteriores, se ha empleado una arquitectura hexagonal en el código, que básicamente desacopla la lógica del sistema para que cada componente sea independiente.

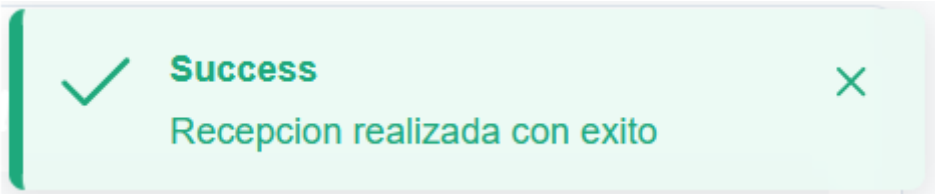
Para lograr esto en nuestro proyecto cada componente se crea dentro de carpetas que definen el tipo de componente:

-En la carpeta **action** se recoge varias componentes que realizan llamadas http POST al BackEnd, como puede ser recibir un objeto o ubicar dicho objeto.
-Una carpeta **admin** que incluye acciones que solo puede realizar el administrador como la administración de usuarios.
-Carpeta **auth** que gestiona el login
-Carpeta **content** que incluye los componentes de vista que realizan solo llamadas http GET a BackEnd

Estas carpetas se encuentran en el path /src/app/presentation/components. Cada carpeta contendrá su routing.module que determinará los caminos que llevaran a cada componente, que dicho routing.module luego se recogen en un módulo de rutas global que es app-routing.module.

Cada componente dispone de su propio módulo, donde se declaran los imports de módulos de componentes que se van a usar, como puede ser passwordModule de primeng/password para los formularios de contraseñas o un módulo que se encuentra casi todos los componentes es el de ToastModule que permite mostrar notificaciones dentro de la aplicación web empleando el servicio de MessageService de primeng/api, para que el usuario reciba un feedback de las acciones que realiza.



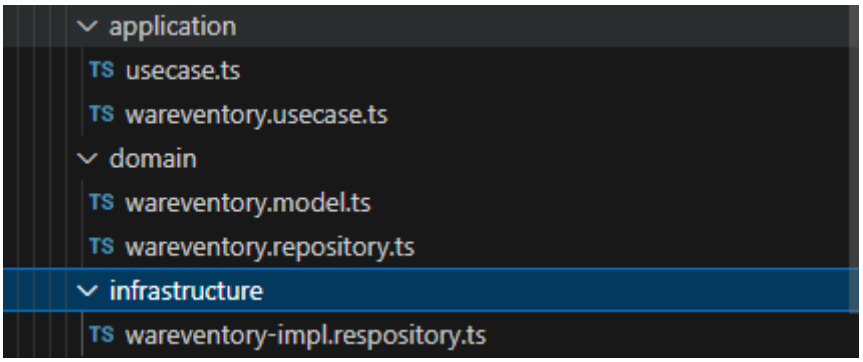


Para poder realizar las operaciones a BackEnd, es decir los puertos y adaptadores descritos, se emplean las siguientes capas que se encuentran en el camino de src/app/features:

-Carpeta **domain:** aquí se encuentra los modelos y repositorios, en este proyecto no se ha usado los modelos para mantener una estructura más flexible, en los repositorios se definen las funciones abstractas que se heredarán en la implementación

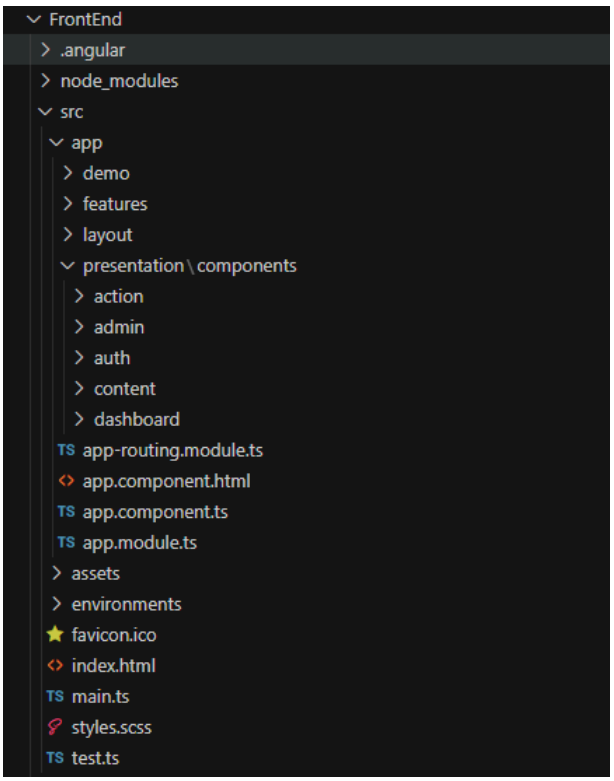
-Carpeta **application**: aquí es donde se encuentra el adaptador que conecta las componentes con la implementación de las funciones del repositorio.

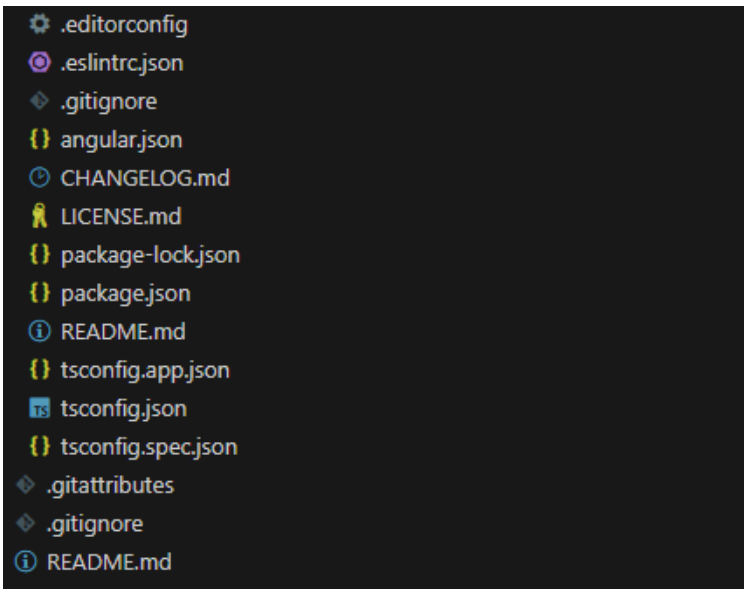
-Carpeta **infrastructure**: aquí se realiza la implementación de las funciones abstractas que básicamente define los headers con los tokens JWT devueltos y los parámetros que se deben de pasar a BackEnd.



Todo esto es recogido para que pueda ser usado por otras componentes en data-module, que define un UseCaseFactory y los proveedores en NgModule, de forma que cualquier componente pueda importar WareventoryUseCase y usar las implementaciones a las llamadas de BackEnd

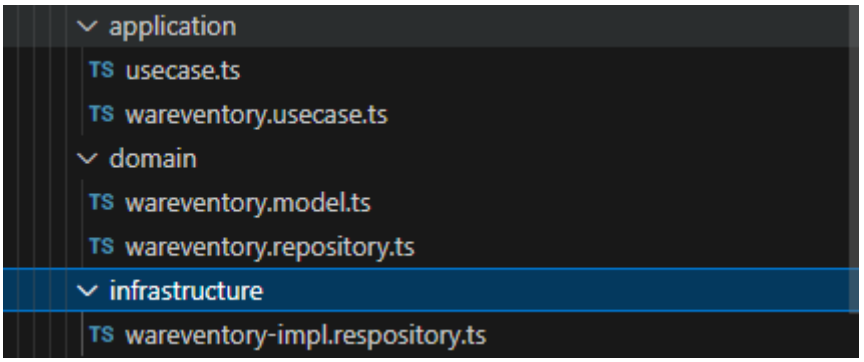
4.3.1 Estructura directorio FrontEnd





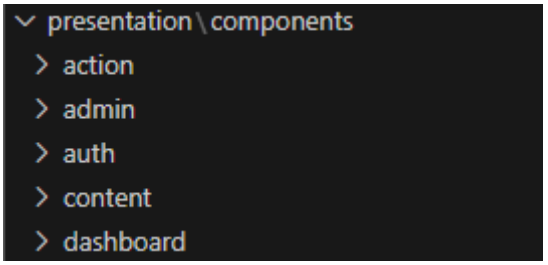
A continuación se va a presentar que contiene cada de las carpetas y los contenidos de los archivos en la estructura del proyecto

- node_modules: contienen las dependencias de librerías del proyecto, aquellas que se instalan haciendo npm install
- src: es la carpeta que contiene el código del proyecto
- app: contiene la implementación de módulos y componentes
- demo: es la carpeta que contiene ejemplos de todas las componentes que proporciona la plantilla de Sakai
- features: aquí se encuentra la implementación de puertos y adaptadores de arquitectura hexagonal explicado anteriormente



- layout: es la carpeta donde se encuentra una copia modificada del layout de demo para ajustar las componentes a nuestro proyecto

- presentation: esta es la carpeta donde la implementación de todas las componentes para el manejo del inventario de nuestra aplicación.
  Que a su vez contiene distintas carpetas para clasificar los diferentes componentes dependiendo de la función que desempeñen.



- Action: se recogen las componentes que realizan una acción sobre la base de datos

- Admin: se recogen las componentes que realizan una acción de administrador

- Auth: que contiene una componente de login junto a un AuthGuard que determina si el usuario puede acceder a ciertos componentes o no.

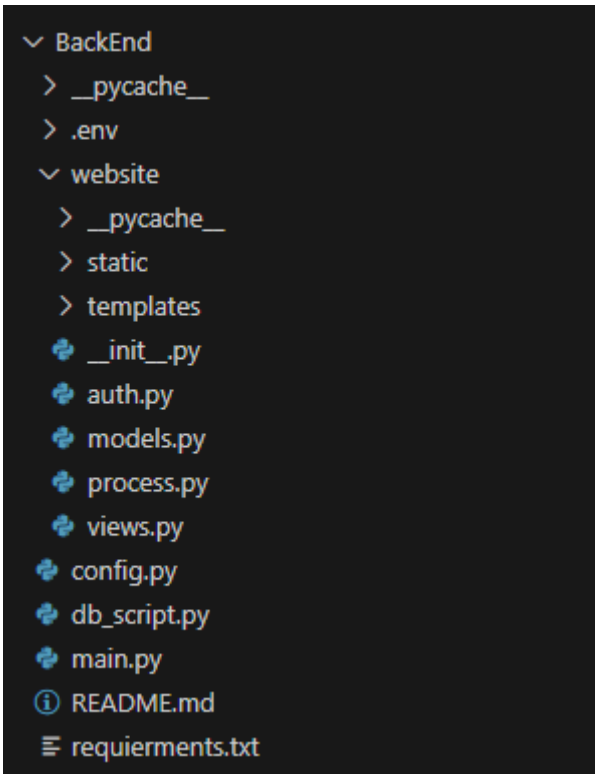
- Content: contiene las componentes que solo realizan peticiones http GET.

- Dashboard: es el componente de landing que realiza una función de recolección de datos estadísticos.

La componente app-routing contiene las rutas a los distintos módulos de rutas que tiene cada componente

4.4 Estructura de BackEnd

Dado que se ha usado el framework de Python-Flask para el desarrollo del backend, la estructura es bastante sencilla.



Se explicará los distintos scripts agrupados en orden de ejecución:

- main.py: es el script que se ejecutara para arrancar la parte servidor del proyecto
- config.py: es una clase que contiene parámetros como es la SECRTE_KEY para la encripción del token JWT, el algoritmo usado, y la url de la base de datos.
- requirements.txt: contiene los paquetes que se deben de instalar con pip install -r ./requierments.txt
- Carpeta .env: este es un entorno virtual creado con python venv .env, que servirá para instalar los paquetes del requirements.txt de forma no global en el sistema, para activar el entorno virtual se ha de ejecutar .env/Scripts/activate
- __init__.py: contiene una funcion llamada create_app() que inicializa configuraciones globales, inicia la conexión con mongoDB y activa la política de CORS para las dos componentes de logica que son process.py y auth.py
- db_script.py: es el script que genera datos de prueba que son insertados en la base de datos.

- models.py: aqui se encuentra la definiciones de todas las clases de la base de datos de mongo en python, utilizando la librería de mongoengine, donde las clases simples se crean usando Document, los arrays dentro de clases se crean usando EmbeddedDocument y las clases dinámicas como son los logs se crean usando DynamicDocument. Cada clase además contiene funciones propias como puede ser get_product_location_by_sku(cls,sku) que se definen usando @classmethod siendo cls una referencia a la propia clase.

- auth.py aqui solo se definie una sola ruta, /login POST, esta ruta se encargará de comprobar los datos del usuario con la base de datos y devolver un token JWT cifrado con nuestra secret_key con la información del token JWT, el nombre del usuario y su rol, y una fecha de expiración de un día

- process.py: este es el script encargado de realizar la gran parte de las conexiones con base de datos, teniendo 15 rutas cada una con distinto métodos que puede aceptar, que son GET, POST, PUT y DELETE, para asegurar que el acceso a dichas rutas esté controlado y verificada para que solo usuarios registrados puedan realizar las operaciones se han creado dos decoradores, @token_requiered y @admin_token_requiered, que comprueban el token disponible en los headers de las peticiones http, que comprueba de decodificación del token con nuestra secret_key y comprueba que el token no haya expirado con jwt.decode().

4.4 Estructura de Sakai Template

Para poder ver las componentes de la plantilla se encuentran en el siguiente path src/app/layout/shared, donde se encuentran los menús, topbar, footer, etc.

Aunque exista una componente llamada SideBar, si se quiere modificar la barra de la izquierda se debe recurrir a la componente llamada app.menu.component, donde los elementos se dentro de la barra se han de definir dentro de una lista. Los archivos CSS se encuentran en el path de src/assets

5. Funcionalidades de la aplicación

En este apartado se procedera a realizar una breve expliación sobre cada una de las funcionalidades implementadas que conforman nuestra apliación:

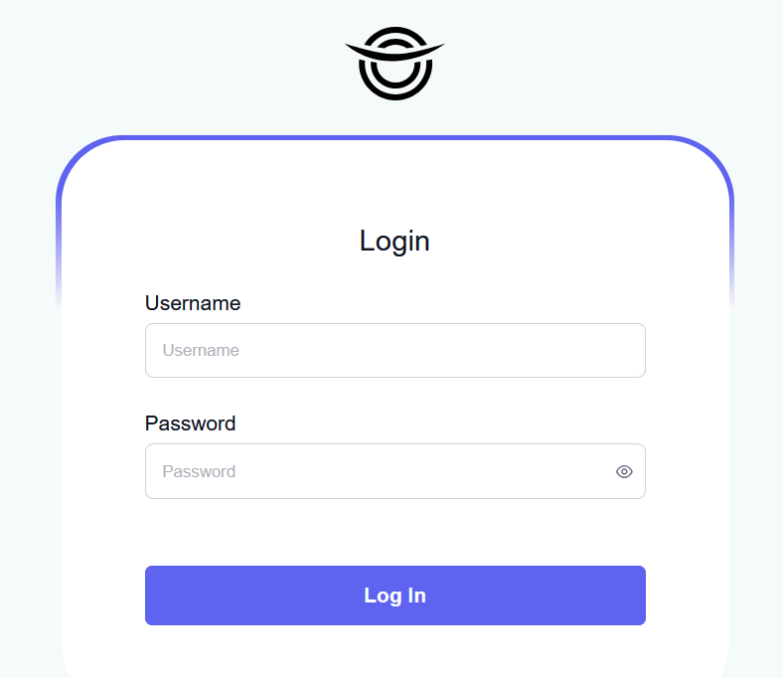
- Gestion de usuarios
- Consultas de inventario
- Acciones de manejo de inventario
- Acciones de administrador
- Traducción

5.1 Gestion de usuarios

Se diferencia dos tipos de usuarios, un usuario con role “basic” y otros usuarios con rol “admin”, la aplicación sólo es accesible a usuarios registrados dentro del sistema, no existe la posibilidad de un usuario anónimo ya que no tiene razón para entrar a un sistema logístico si no pertenece a dicho sistema.

5.1.1 Iniciar sesión

Al inicio de la aplicación, si no se ha conservado un token con fecha de expiración válida en el navegador, se transferirá al usuario automáticamente a una página de inicio de sesión, donde deberá introducir las credenciales proporcionadas por un administrador.

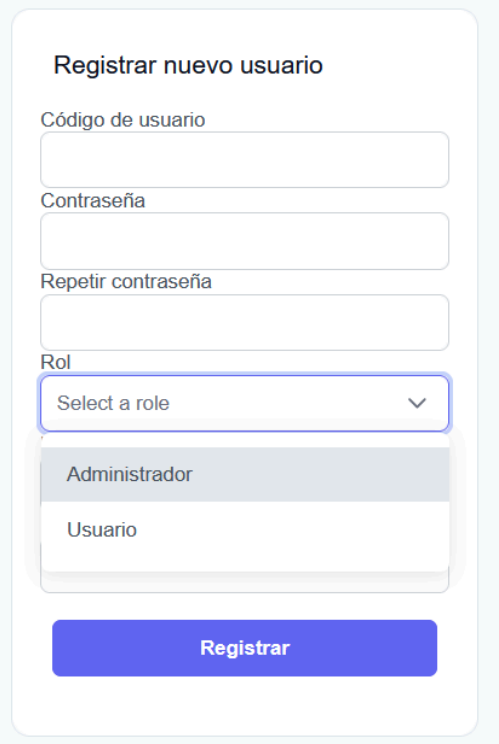


5.1.2 Cerrar sesión

Una vez iniciada sesión, se guarda un token JWT dentro del navegador, si se quiere cerrar sesión, existe un botón en la parte superior derecha que cierra la conexión.

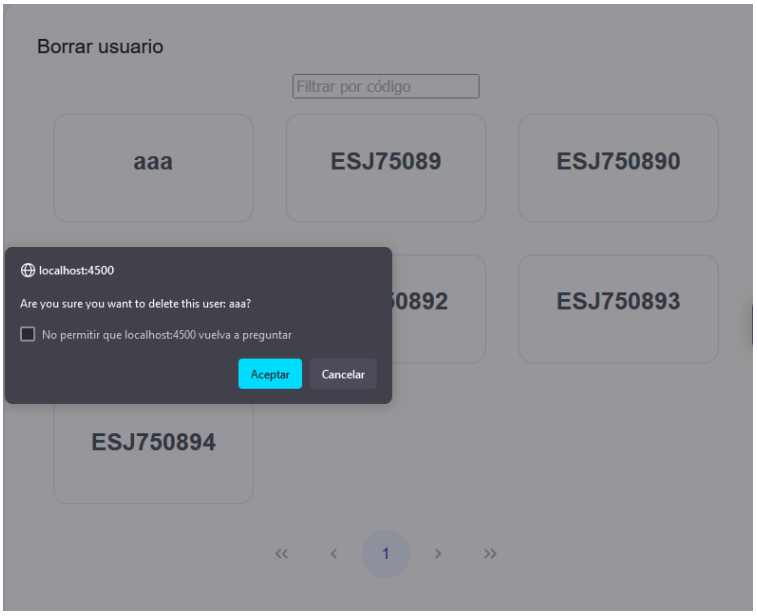
5.1.3 Registrar usuario

Como se ha explicado anteriormente, este es un sistema cerrado, y solo los administradores pueden crear nuevos usuarios, y se les asigna un rol básico o de administrador al perfil nuevo.



5.1.4 Eliminar usario

Esta también es una funcionalidad exclusiva de los administradores, se les mostrará una tabla con páginas de todos los códigos de usuarios existentes en el sistema y se podrá borrar los usuarios después de confirmar una alerta de confirmación. También dispone de una barra de búsqueda que devuelve la lista de códigos que contienen la cadena de texto introducida.

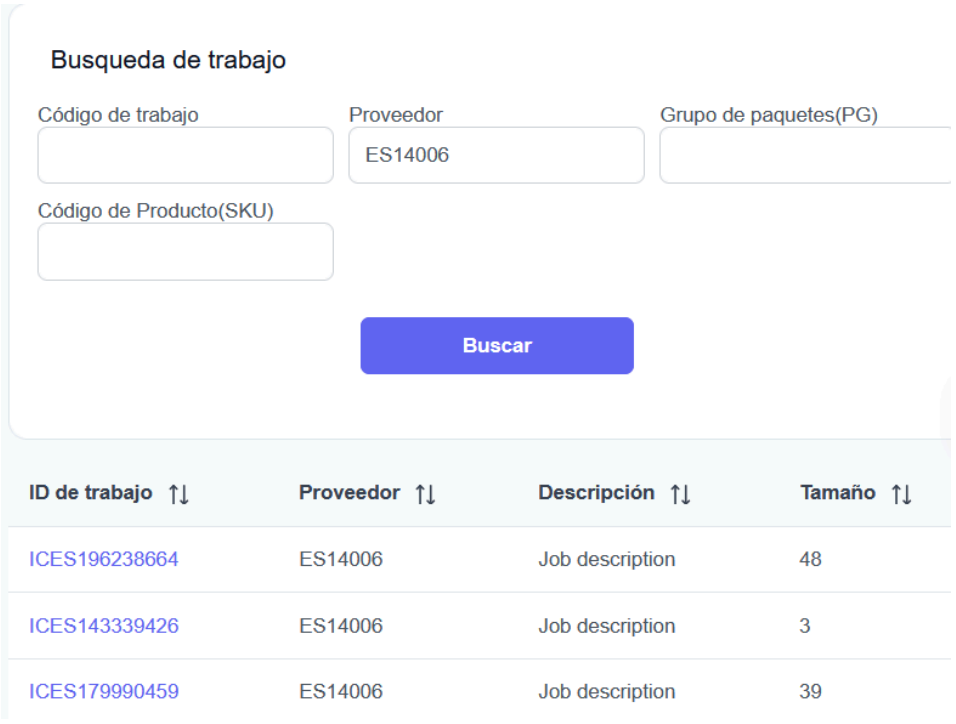


5.2 Consultas de Inventario

Esta es una funcionalidad que permite a los usuarios ver datos del sistema logístico, tienen acceso todos los usuario registrados en el sistema, usando el token JWT para autentificarse en la conexión a BackEnd.

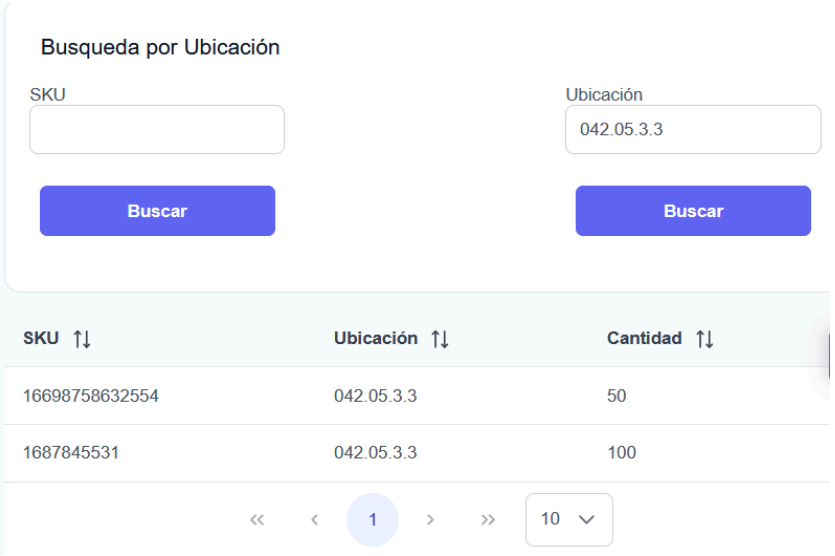
5.2.1 Vista de trabajos

El usuario puede consultar los trabajos que se encuentran en el sistema a través de varios parámetros, el propio código del trabajo, el proveedor, un PG(package group) que se encuentre dentro del trabajo, o un SKU de un objeto que se encuentre en un trabajo, esto se realiza con una query dinámica, donde se puede introducir entre 1-4 parámetros para realizar el filtro de datos.



5.2.2 Búsqueda en inventario

El usuario puede realizar búsqueda del inventario de almacén, es decir puede ver en qué ubicaciones se encuentra un objeto o qué objetos se encuentra en una ubicación.



5.3 Acciones de manejo de inventario

En este apartado se procederá a explicar, las funcionalidades relacionadas al flujo completo que siguen los almacenes logísticos en cuanto a la entrada, manejo y salida de productos.

5.3.1 Recibir

Cuando llega ya sea un contenedor, camión o otro medio de transporte cargado con mercancía, los bienes se descargan en los almacenes y se registran su llegada en el sistema en este acto que es la recepción de paquetes.

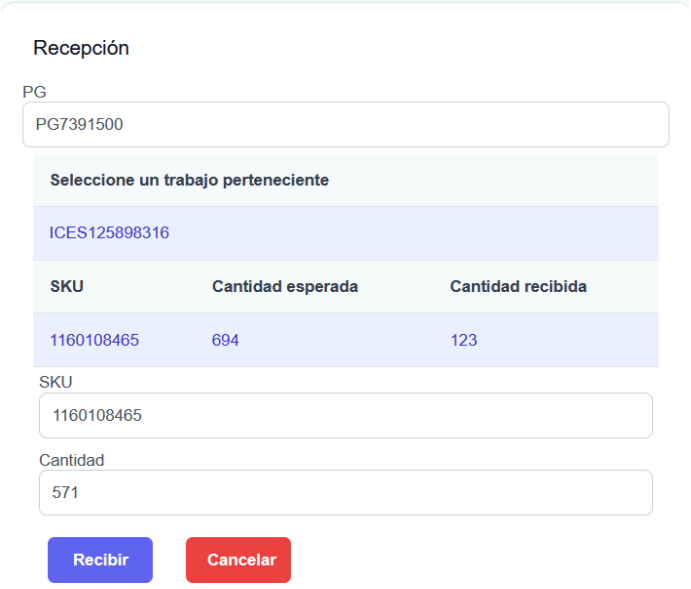
Para realizar una recepción, en casos habituales se tiene una pegatina que incluye un código que identifica la caja, y un SKU que identifica los objetos, ambos representados con texto y código de barras, vea la siguiente imagen como referencia.

Antes de la llegada de los bienes, se tendrá ya disponible el trabajo que contiene dicha caja junto a otras más que vengan incluidas en el mismo trabajo.

Para recibir una caja se ha de escanear el código del paquete, y se preguntará a qué trabajo pertenece, por que existen casos en los que se puede repetir el PG (package group) en trabajos diferentes.

Después se preguntará que SKU se desea recibir, ya que también existen casos donde una caja contiene varios productos diferentes, por último, se introducirá la cantidad que se recibe, dado que el trabajo registra la cantidad total de bienes, pero este puede venir separada en varias cajas.

Con todos los parámetros anteriores introducidos, se hará una recepción correcta de una caja.



Si en algún caso, las cajas no vienen con pegatinas pero los objetos si, se ha hecho una funcionalidad extra de recepción directa, donde solo se introduce el SKU del objeto y la cantidad. Este caso es raro pero si pasa a veces en los almacenes, donde las cajas vienen estropeadas, y se ha de reportar a los líderes de grupo o a nivel de administración antes de proceder con las recepciones.

5.3.2 Ubicar

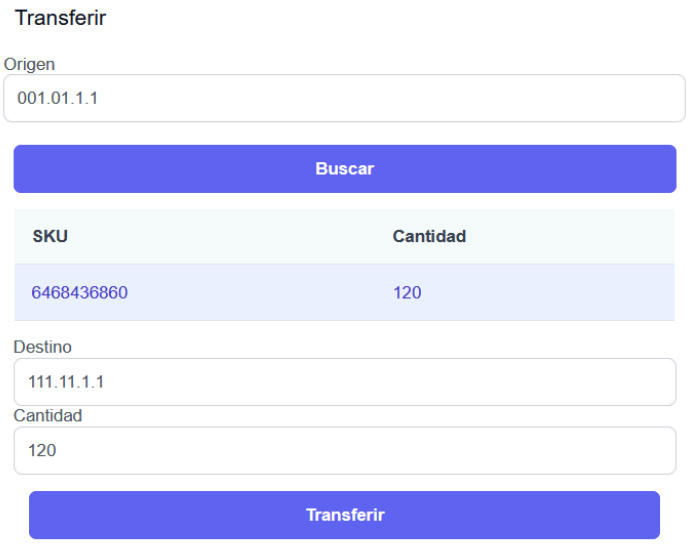
Una vez se hayan recibido los paquetes, se podrá ubicar los objetos en las diferentes ubicaciones que tenga el lugar, puede ser estanterías normales o de altura, palets, etc.

Los objetos recibidos se almacenarán en una colección de la base de datos llamada Reception Basket, para ubicar, se escaneara el código del producto (SKU), y se mostrara las diferentes recepciones relacionadas a dicho objeto, se elige el que corresponde y se introduce una ubicacion, las ubicaciones en los almacenes suelen tener un código de barras para escanearlas también.

5.3.3 Transferir

En el caso que se quiera reubicar un objeto ya almacenado, se tendrá la opción de transferir, se puede realizar introduciendo la ubicación de origen, seleccionando el objeto que se quiera transferir, el destino y la cantidad que se vaya a relocalizar.

Esto es útil ya que a lo largo de las operaciones de almacenamiento de los objetos, un mismo producto puede estar repartido por varias ubicaciones diferentes ocupando sitio y reduciendo eficiencia, con esto se puede juntar los objetos o almacenar objetos en lugares más apropiados.

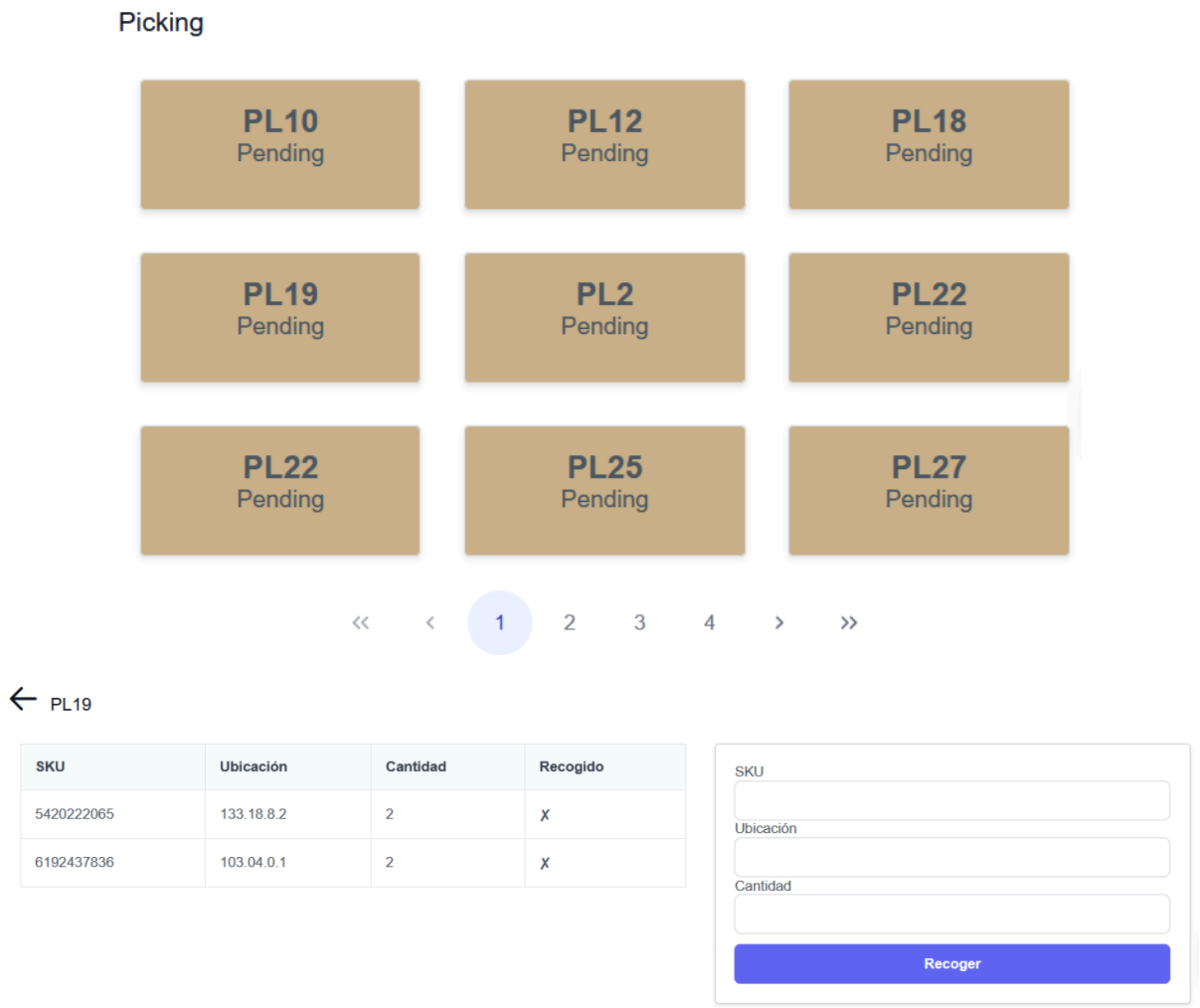


5.3.4 Picking

Esta operación se realiza en base a una lista de trabajos de envíos que se quieren realizar, existen almacenes con equipos especializados en recorrer las naves logísticas, recogiendo todos los objetos que se encuentren apuntados en las listas proporcionadas.

Para realizar un picking, el usuario escoge una de las listas proporcionadas, y escanea los objetos, su origen e introduce la cantidad. Si se completa la lista se mostrará la opción de terminar lista y podrá seguir con otra lista.

En los almacenes, los objetos se suelen almacenar en bandejas con el código de lista correspondiente, para que los usuarios que realizan el empaquetado y envío recojan los objetos de dicha bandeja.

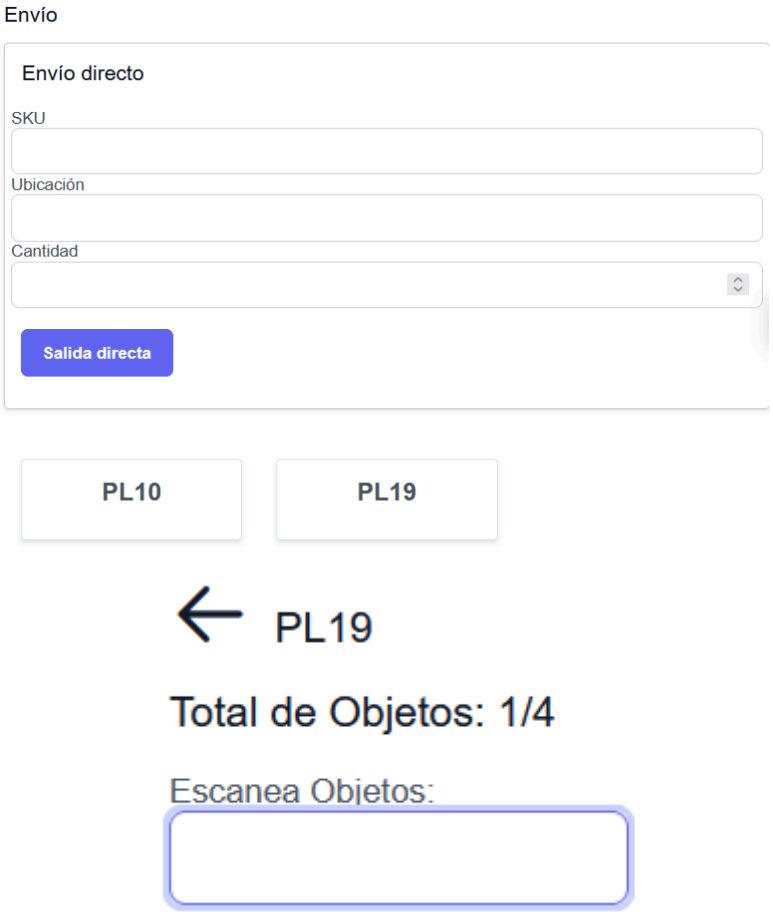


5.3.5 Shipping

Por último, los objetos recogidos se deben de empaquetar y enviar, donde se escaneara los objetos de las lista a ciegas, es decir, se mostrará la cantidad total de objetos de la lista, y el usuario deberá escanear los SKU de los objetos, en cuanto se complete la cantidad es cuando se ha realizado un empaquetado y envío completo.

Se realiza de esta forma para asegurar que los objetos que se envían son realmente los recogidos en la lista, y que el usuario no pueda realizar envíos introduciendo los códigos a mano si le falta un objeto desconocido.

Los usuarios de empaquetado y envío realizan las operaciones sacando los objetos de las bandejas con el código de trabajo correspondiente. En el caso de que falte objetos o no coincidan se reporta a niveles superiores y se ven los registros de actividad.



Se ha implementado también, una funcionalidad de envío directo, que recoge un objeto directamente de la ubicación y los envía, esto servirá en caso de que existan pedidos urgentes, pues que se pueda realizar la acción de recoger y enviar en una sola acción.

5.4 Acciones de administrador

El administrador se encargará de registrar y eliminar usuarios del sistema, además de administrar las entradas, inventario y salidas.

El administrador tiene un menú que solo se mostrará si es un administrador, en el caso de que manipulando el almacenamiento del navegador se pueda acceder a este menú, no puede tampoco realizar las acciones debido al decorador de @admin_token_requiered en BackEnd.

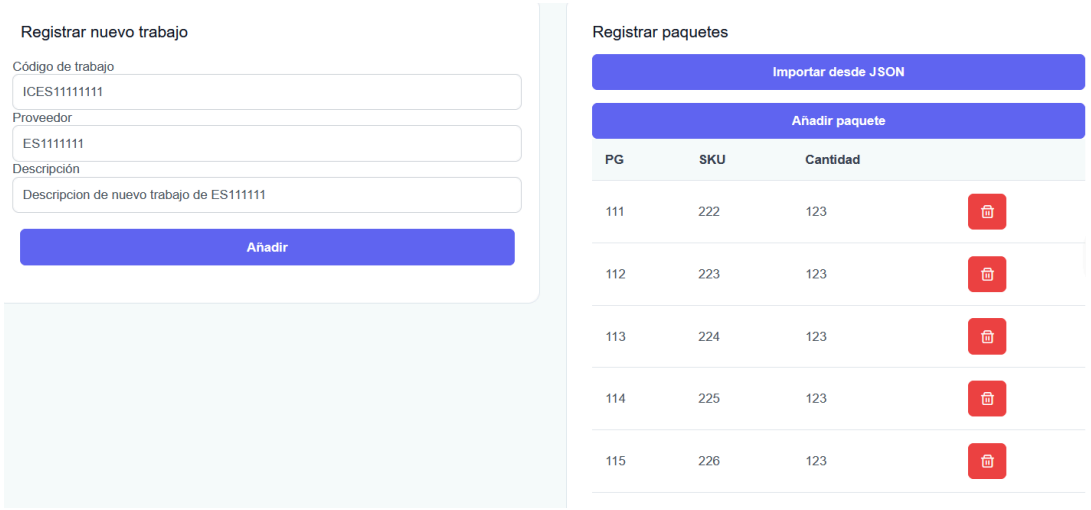
5.4.1 Añadir un nuevo trabajo

Para realizar las recepciones mencionadas anteriormente, se ha de disponer la lista con los paquetes que van a entrar, para realizar esto, el administrador le llegará un pedido con los datos del trabajo, y los deberá de introducir en la base de datos.

Deberá de introducir un código de trabajo, el código del proveedor, y una descripción, con esto crea un trabajo nuevo, pero no tiene ningún paquete dentro del trabajo, para añadirlos, se puede realizar de dos formas, introduciendo los paquetes manualmente con con los PG, SKU y cantidad o a través de un JSON array, siguiendo el siguiente formato:

[{“PG”:”nombrePG”,”SKU”:”SKU nombre”,”quantity”:100 },
{“PG”:”nombrePG”,”SKU”:”SKU nombre”,”quantity”:100 },
……]

Una vez añadido los trabajos, son inmutables, es decir, que no se podra modificar sobre los códigos de paquetes, objetos ni la cantidad esperada, las cantidades recibidas y ubicadas se actualizarán de forma automáticamente, cuando se realice dichas acciones.

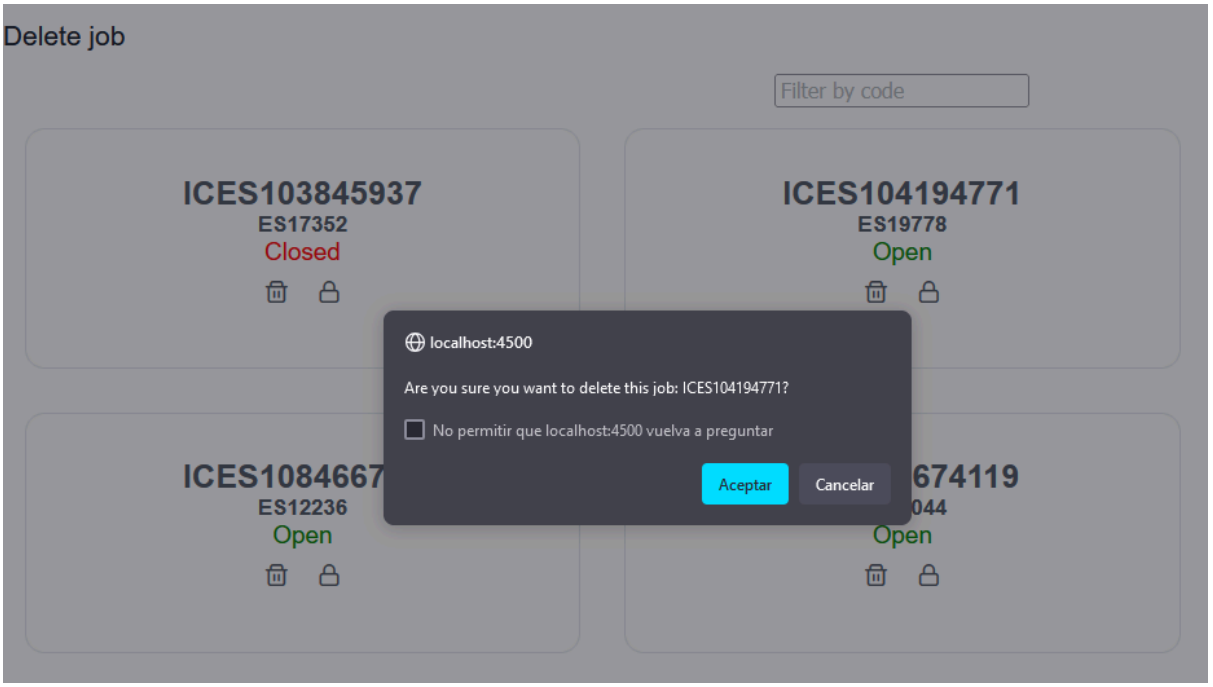


5.4.2 Administrar trabajos

El administrador también podrá borrar trabajos o cerrar los trabajos, antes de realizar alguna de las operaciones anteriores, le saltará una ventana de alerta de confirmación.

Los administradores deben de tener especial cuidado al eliminar un trabajo, ya que se pierden muchos datos aunque se hagan backups de las bases de datos, también que es una operación poco común ya que se los trabajos también se usan como historial de los objetos del almacén.

En cuanto a “cerrar” los trabajos, esta acción se realiza cuando un trabajo no es urgente y se decide redirigir recursos a trabajos con más urgencia, o cuando un trabajo no se ha trabajado durante mucho tiempo, donde se debe de realizar una comunicación con los clientes o los proveedores para ver la situación de que hacer con las mercancías.



5.4.3 Administrar listas de Recogida/Envío

Para que los usuarios básicos puedan realizar una recogida de objetos para luego ser empaquetados y enviados,el administrador tendrá que crear listas con los objetos de cada pedido, el administrador deberá de introducir un código para identificar la lista, y al igual que los paquetes de los trabajos, se podrán crear uno a uno o introduciendo un JSON array con el siguiente formato:

[{"SKU":"111","quantity":111},{"SKU":"222","quantity":222},.....]

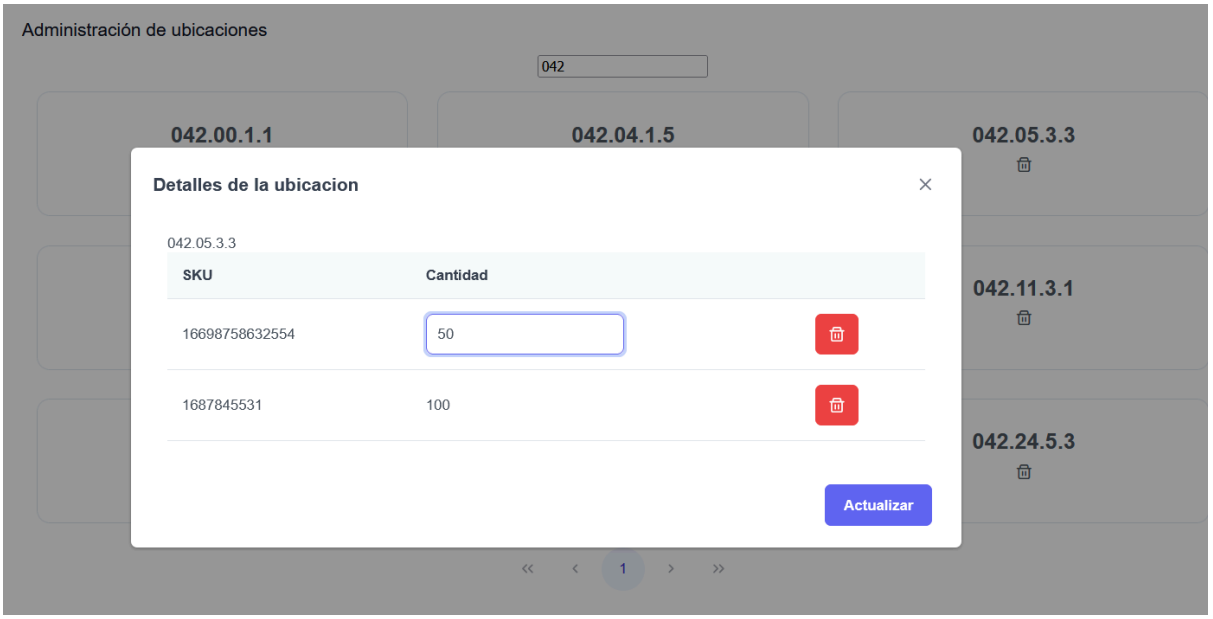
Los objetos añadidos, una vez pasen a BackEnd, se buscará una localización que contenga dicho objeto, y se le asignará la localización en la lista para que el usuario que recoja los objetos los pueda ubicar fácilmente.

También se tiene la opción de borrar una lista después de confirmar la operación en una ventana de alerta.

5.4.4 Administrar ubicaciones

Por último, el administrador tiene la posibilidad de modificar directamente sobre las ubicaciones disponibles, esto incluye eliminar la ubicación por completo, o modificar sobre los objetos existentes en la localización.

Si se borra la ubicación también desaparecerán los objetos almacenados en la ubicación, si se selecciona una ubicación, se podrá editar sobre los elementos o eliminar dichos elementos, modificando su código o cantidad



Esto es de especial utilidad, ya que a lo largo del ciclo de vida de los almacenes, es inevitable que se pierdan algunos objetos, o que ocurran errores en el flujo de trabajo, ya sea en la parte de recepción, ubicación, etc.

En algunos casos especiales, se manda a personas a contar cuántos objetos hay en algunas ubicaciones y se debe de actualizar en la base de datos, e informar de la diferencia de datos con los clientes o proveedores.

5.5 Traducciones

Como funcionalidad extra, se han realizado las traducciones siguiendo el formato i18n, donde se han creado varios archivos JSON para almacenar las traducciones de forma local, que son más precisas y modificables.

Se ha empleado TranslateService de @ngx-translate/core que es un servicio proporcionado por ngx, donde se usan los códigos de idiomas de BCP 47, que define códigos como “es” para español, “fr” para francés, etc.

Se han realizado traducciones al Español, Inglés, Alemán, y Chino, para añadir nuevas traducciones se realiza de manera muy sencilla siguiendo los siguientes pasos:

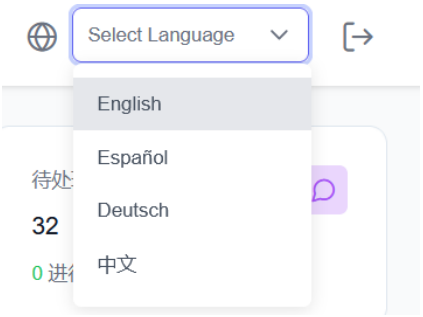
-Se ha de crear un nuevo archivo con XX.json en la carpeta i18n en el siguiente path: /FrontEnd/src/assets/i18n, donde XX es el código del idioma siguiendo es estandard BCP47

-Por último, se ha de de añadir una nueva opción en la variable de languages en el archivo de app.topbar.component.ts disponible en el path: /FrontEnd/src/app/layout/shared

```
public languages: any[] = [
    { label: 'English', value: 'en' },
    { label: 'Español', value: 'es' },
    { label: 'Deutsch', value: 'de' },
    { label: '中文', value:'zh-cn'},

];
```

Por último en la parte superior derecha, se encuentra un botón con un icono de planeta que permitirá al usuario cambiar de idioma.



Las traducciones se han hecho de forma manual, excepto el alemán donde se ha realizado usando un traductor, los archivos JSON son revisables y modificables con una gran sencillez en el caso de que alguna traducción no se correcta y revisado por un hablante de la lengua traducid

6. Conclusiones y trabajo futuro

6.1 Conclusiones del trabajo

Durante la realización de este proyecto se ha aprendido bastantes conceptos y tecnologías relacionadas al desarrollo web, que en resumen han sido Angular, arquitectura hexagonal, flask, y mongo, que creo que es una combinación bastante inusual de lo normal en cuanto a la combinación de frameworks usados para el desarrollo de una aplicación web.

Se ha realizado una visión inicial de los objetivos planteados inicialmente, que es un sistema básico de administración de inventario de almacenes logísticos, que la idea original vino del sistema usado en un trabajo de verano en los almacenes de 4PX. Aplicando una visión fragmentado en componentes, que permite realizar cada componente sin dependencias gracias a la arquitectura hexagonal, cuyo concepto fue aprendido en los trabajos de prácticas en empresa en el Banco de España.

Se concluye una realización exitosa de los objetivos iniciales, y haber aprendido bastante sobre el proceso de desarrollo web.

6.2 Limitaciones

Dado que es un intento de imitación de un sistema real y maduro, y de no disponer de contactos dentro del sector logístico, al final no se ha podido consultar las diferentes formatos, no se ha podido obtener ejemplos reales y se ha basado el proyecto en cuanto a memoria y experiencia personal.

Al no conocer los distintos formatos como los patrones que deben de seguir los SKU, dado que en internet se encuentran cada página que indica un SKU con formato diferente, y que los Package Group tampoco se ha encontrado información,

posiblemente que solo fuese usado por dicha empresa donde se ha realizado el trabajo de verano, no se ha impuesto un formato fijo para los datos.

Otra limitación ha sido la falta de tiempo para probar procesos reales como podría haber sido, emplear la API de DHL o UPS, empresas de envío para realizar el envío final.

Una de las ideas iniciales fue, hacer un sistema donde se usen las dimensiones de los objetos para realizar una forma optimizada de almacenar objetos empleando inteligencia artificial, pero no se ha realizado por falta de experiencia en el campo de la inteligencia artificial y la falta de tiempo.

6.3 Trabajo Futuro

Durante el desarrollo del trabajo se ha acumulado una cantidad de “Technical-Debt”, estos han sido el la fragmentación de la arquitectura hexagonal, donde se ha usado solo un puerto-adaptador, siendo WareventoryUseCase, un trabajo futuro para reducir la acumulacion de codigo en un solo archivo sería dividir dicho Use Case en varios archivos más pequeños, como podrían ser WareventoryContentUseCase para las peticiones http GET, y de forma similar para las otras peticiones o dividirlos por funciones como basic y admin.

Realizar las conexiones con BackEnd empleando OpenApi generator, con un archivo JSON definiendo todas las peticiones y respuestas. Empleando esta tecnología se consigue crear una conexión más robusta comparada con la que se encuentra actualmente implementada.

Comprender mejor la arquitectura de la plantilla de Sakai, ya que mucho peso de las carpetas del proyecto son los archivos CSS, que se pueden borrar bastante código de dichos archivos que no se usan.

Mejora en general del código, comprobando más datos y excepciones, implementar los modelos de arquitectura hexagonal para una respuesta unificada con modelo de datos.

Conclusions and future work

Conclusions of the work

During the realization of this project the student has learned quite a few concepts and technologies related to web development, which in summary have been Angular, hexagonal architecture, flask, and mongo, which I think is a rather unusual combination of frameworks to be used for the development of a web application.

The initial vision of the objectives has been realized, which is a basic inventory management system for logistics warehouses, where the original idea came from the system used in a summer job at 4PX warehouses. Applying a fragmented vision onto the components, which allowed us to work on each component without dependencies thanks to the hexagonal architecture, whose concept was learnt in the internship work at the Bank of Spain.

The conclusion is a successful achievement of the initial objectives, and to have learned a lot about the web development process.

Limitations

Given that this is an attempt to imitate a real and mature system, and that the student does not have contacts in the logistics sector, in the end we have not been able to obtain real examples and the project has been based on memory and personal experience.

Since we did not know the different formats such as the patterns that the SKUs should follow, given that on the internet there is each page that indicates a SKU with a different format, and that we could not find any information about the Package Group either, it is possible that it was only used by the company where the summer work was carried out, with those limitations in mind, we did not impose a fixed format for the data.

Another limitation was the lack of time to test real processes, such as using the API of DHL or UPS, shipping companies to make the final shipment.

Another one of the initial ideas was to make a system where the dimensions of the objects are used to create an optimized way of storing objects using artificial intelligence, but this has not been done due to lack of experience in the field of artificial intelligence and lack of time.

Future Work

During the development of the work a lot of ‘Technical-Debt’ has been accumulated, one of those has been the fragmentation of the hexagonal architecture, where only one port-adapter has been used, being WareventoryUseCase, a future work to reduce the accumulation of code in a single file would be to divide this UseCase in several smaller files, as could be WareventoryContentUseCase for the http GET requests, and similarly for the other requests or divide them by functionality like basic and admin.

BackEnd connections using OpenApi generator, with a JSON file defining all requests and responses. Using this technology creates a more robust connection compared to what is currently implemented.

Better understanding of the architecture of the Sakai template, as a lot of the weight of the project folders are CSS files, which can delete quite a lot of code from those files that are not used.

General code improvement, checking more data and exceptions, implementing the hexagonal architecture models for a unified response with data model.

Bibliografia

[1]Introducción Arquitectura Hexagonal:
https://codigoencasa.com/arquitectura-hexagonal-con-angular/

[2]Introducción Arquitectura Hexagonal 2:
https://medium.com/ssense-tech/hexagonal-architecture-there-are-always-two-sides-to-every-story-bc0780ed7d9c

[3]Primeng:
https://primeng.org/installation

[4]Video introducción a Arquitectura Hexagonal:
Aplicando Clean Architecture en un proyecto Angular

[5]Lista Video introduccion Flask-MongoEngine:
How to connect to mongo from mongoengine in python - Python mongoengine …

[6]Video ejemplo traducciones i18n:
Angular 16 Tutorial - Internationalization (i18n) with Json Data #16

[7]JWT:
https://en.wikipedia.org/wiki/JSON_Web_Token

[8]Sakai Template:
https://github.com/primefaces/sakai-ng

[9]Angular AuthGuard:
https://medium.com/@jaydeepvpatil225/auth-guards-in-angular-6960950b3c6c

[10] Flask JWT Token validate
https://stackoverflow.com/questions/32510290/how-do-you-implement-token-authentication-in-flask

[11]Python virtual environment
https://docs.python.org/3/library/venv.html

APÉNDICES

Apéndice A - Repositorios

Se adjuntan el enlace al repositorio en Github:

https://github.com/hdswa/Wareventory