UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Aplicación móvil para el control de producción en tiempo real en una fábrica textil

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Ferrándiz Santamaría, Jairo

Tutor/a: Esparza Peidro, Javier

CURSO ACADÉMICO: 2022/2023

Resumen

Este trabajo de final de grado consiste en el desarrollo de una aplicación móvil utilizando Flutter y Flask para controlar los artículos en producción en una fábrica textil. La aplicación trabaja en tiempo real y proporciona información sobre los artículos pendientes de realización, utilizando una base de datos de la empresa. Además, la aplicación es capaz de informar a los gerentes sobre el estado de la producción en cualquier momento. Esta herramienta puede ser de gran utilidad para optimizar y mejorar la eficiencia en la gestión de la producción en la fábrica textil.

Palabras clave: Aplicación móvil, Control de producción, Tiempo real, Fábrica textil, Artículos en producción, Base de datos, Flutter, Flask, Gerentes, Estado de la producción, Optimización, Eficiencia, Gestión de la producción, Producción industrial, Sistemas de información, Desarrollo de aplicaciones, Flujos de trabajo, Monitoreo de procesos, Análisis de datos, Herramientas de gestión empresarial.

Abstract

This bachelor's final project consists of the development of a mobile application using Flutter and Flask to control the items in production in a textile factory. The application works in real time and provides information about the items pending completion, using a company database. In addition, the application is able to inform managers about the status of production at any time. This tool can be of great use to optimise and improve the efficiency of production management in the textile factory.

Key words: Mobile application, Production control, Real time, Textile factory, Articles in production, Database, Flutter, Flask, Managers, Production status, Optimisation, Efficiency, Production management, Industrial production, Information systems, Application development, Workflows, Process monitoring, Data analysis, Business management tools.

Índice

1. Introducción 8

1.1 Motivación 9

1.2 Estudio del mercado 9

1.3 Objetivos del proyecto 10

1.4 Estructura del resto de la memoria 10

2. Análisis del problema 11

2.1 Acolchado 12

2.2 Corte 13

2.3 Corte-Luís 13

2.4 Empaquetado 14

2.5 Almacén 14

3. Solución del problema 16

3.1 Arquitectura de la aplicación 16

3.2 Tecnologías utilizadas 18

3.3 Capa de presentación de datos 19

3.3.1 Acolchado 19

3.3.2 Corte 20

3.3.3 Corte-Luís 21

3.3.4 Empaquetado 21

3.3.5 Almacén 21

3.4 Capa de lógica de negocio 27

3.5 Capa de acceso a datos 31

4. Tour por la aplicación 34

4.1 Sección de Acolchado 35

4.2 Sección de Corte 37

4.3 Sección de Empaquetado 39

4.4 Sección de Almacén 40

5. Conclusiones y trabajos futuros 44

6. Bibliografía 46

Índice de figuras

Figura 1: Diagrama de flujo sobre el funcionamiento de la aplicación 16

Figura 2: *Diagrama de bloques sencillo que muestra las 3 capas de la aplicación* 18

Figura 3: Captura de pantalla con la distribución de los archivos .dart 22

Figura 4: Diagrama de flujo explicativo sobre cómo actúan los ficheros .dart 23

Figura 5: Mockup de la pantalla de inicio de la aplicación 24

Figura 6: Mockup de la producción donde se muestra el GridView de selección de modelo .. 24

Figura 7: Mockup de la producción donde se muestra selector de color y la interacción del empleado 24

Figura 8: Mockup de la producción donde se muestra el Drawer con la selección de medidas y la confirmación 24

Figura 9: Mockup del almacén donde se muestra la pantalla principal a la cual vuelve tras cada envío 25

Figura 10: Mockup del almacén donde se muestra el MultiSelectDialogField de selección clientes 25

Figura 11: Mockup del almacén donde se muestra el MultiSelectDialogField de selección productos 25

Figura 12: Mockup del almacén donde se muestra el Drawer para visualizar las secciones a controlar 25

Figura 13: Mockup del almacén donde se muestra una de las secciones controladas visualizando que se está haciendo 25

Figura 14: Fragmento de código perteneciente a la capa de presentación de datos 26

Figura 15: Fragmento de código perteneciente a la capa de lógica de negocio 29

Figura 16: Tabla con todas las rutas publicadas en el servicio RESTful con una breve explicación sobre su comportamiento 30

Figura 17: Diagrama de flujo explicativo sobre cómo actúan los métodos de la aplicación desplegada en Flask (app.py) 31

Figura 18: Diagrama ER de la base de datos de la empresa 32

Figura 19: Diagrama ER de la base de datos de la empresa 33

Figura 20: Captura de la aplicación que muestra el login 34

Figura 21: Captura de la aplicación que muestra el teclado que muestra el login 34

Figura 22: Captura de la aplicación que muestra la selección de medidas en la sección de Acolchado 35

Figura 23: Captura de la aplicación que muestra la selección de familia en la sección de Acolchado 35

Figura 24: Captura de la aplicación que muestra la selección de modelo en la sección de Acolchado 35

Figura 25: Captura de la aplicación que muestra la selección de color y el posterior desplegable informativo en la sección de Acolchado 35

Figura 26: Captura de la aplicación que muestra la confirmación del producto en la sección de Acolchado 36

Figura 27: Captura de la aplicación que muestra la cantidad de productos que debe realizar el empleado de la sección de Acolchado 36

Figura 28: Captura de la aplicación que muestra la selección de línea en la sección de Corte 37

Figura 29: Captura de la aplicación que muestra la selección de familia en la sección de Corte 37

Figura 30: Captura de la aplicación que muestra la selección de modelo y color en la sección de Corte 37

Figura 31: Captura de la aplicación que muestra la confirmación del producto en la sección de Corte, con el aviso de “Revisa el desplegable” 37

Figura 32: Captura de la aplicación que muestra la confirmación del producto en la sección de Corte desde el desplegable 38

Figura 33: Captura de la aplicación que muestra como se ha redirigido al trabajador al producto con el mismo modelo y color para que así pueda trabajar de una manera más optima ... 38

Figura 34: Captura de la aplicación que muestra la disposición de la sección de Empaquetado 39

Figura 35: Captura de la aplicación que muestra el escáner de la sección de Empaquetado . 39

Figura 36: Captura de la aplicación que muestra la lista resultante del escaneo en la sección de Empaquetado 39

Figura 37: Captura de la aplicación que muestra la confirmación del producto en la sección de Empaquetado 39

Figura 38: Captura de la aplicación que muestra la disposición de la sección de Almacén 40

Figura 39: Captura de la aplicación que muestra la selección de clientes en la sección de Almacén 41

Figura 40: Captura de la aplicación que muestra la selección de productos en la sección de Almacén 41

Figura 41: Captura de la aplicación que muestra la selección de cliente y producto listos para el albarán en la sección de Almacén 41

Figura 42: Captura de la aplicación que muestra la confirmación del albarán en la sección de Almacén 41

Figura 43: Captura de la aplicación que muestra el desplegable de control de las secciones en la sección de Almacén 42

Figura 44: Captura de la aplicación que muestra los artículos pendientes de producir de la sección de Empaquetado 42

Figura 45: Captura de la base de datos de Producción donde se muestran diversos albaranes 43

1. Introducción

En la actualidad, la eficiencia en la gestión de la producción es un factor clave para el éxito de cualquier empresa, especialmente en el ámbito de la fabricación y producción de bienes. Con el objetivo de mejorar la gestión y optimizar el proceso productivo, en este trabajo de final de grado se ha desarrollado una aplicación móvil utilizando Flutter y Flask para el control de los artículos en producción en una fábrica textil. La aplicación, que trabaja en tiempo real y utiliza una base de datos de la empresa, proporciona información sobre los artículos pendientes de realización y es capaz de informar a los gerentes sobre el estado de la producción en cualquier momento. Esta herramienta puede ser de gran utilidad para optimizar y mejorar la eficiencia en la gestión de la producción en la fábrica textil, y servir de ejemplo para futuros proyectos similares en otras empresas del sector. En este trabajo se detallará todo el proceso de análisis, diseño y desarrollo de la aplicación, así como sus funcionalidades y resultados obtenidos.

1.1 Motivación

La industria textil es un sector en constante evolución que requiere de una gran eficiencia en su proceso productivo. En este contexto, la gestión del control de producción es esencial para conseguir un rendimiento óptimo y asegurar la satisfacción del cliente. Sin embargo, en muchas fábricas textiles, el control de producción se realiza de manera manual, lo que puede provocar errores, retrasos y pérdida de información importante.

En este sentido, el objetivo de este Trabajo de Fin de Grado es desarrollar una aplicación móvil que permita controlar los artículos en producción en tiempo real y mejorar la eficiencia en la gestión de la producción en una fábrica textil. Esta aplicación será desarrollada utilizando las tecnologías de Flutter y Flask, que permitirán una alta eficiencia y flexibilidad en la gestión de datos y comunicación con el sistema central de la empresa.

Esta aplicación se considera especialmente relevante en un contexto en el que la tecnología móvil se ha convertido en una herramienta esencial en el día a día de cualquier persona y en cualquier sector productivo. Además, el uso de tecnología avanzada puede proporcionar ventajas competitivas a la empresa, ya que permitirá una gestión más eficiente y una reducción de costes. En este sentido, este Trabajo de Fin de Grado supone un paso más en la evolución de la industria textil hacia la digitalización y la mejora de la eficiencia en su proceso productivo.

1.2 Estudio del mercado

En el estudio de mercado realizado para este proyecto se ha encontrado que existen aplicaciones similares enfocadas en el control de producción en fábricas, sin embargo, ninguna de ellas cubre todas las necesidades y funcionalidades requeridas por la empresa en cuestión.

La mayoría de las aplicaciones existentes en el mercado se centran en el seguimiento de la producción de manera general, sin ofrecer la posibilidad de visualizar los detalles de cada uno de los artículos producidos. Por otro lado, algunas aplicaciones sí permiten el seguimiento de los detalles de producción, pero carecen de la capacidad de informar a los gerentes en tiempo real sobre el estado de producción de la fábrica.

Además, se ha observado que muchas de las aplicaciones existentes son costosas y complicadas de usar, lo que hace que su adopción sea difícil para empresas de menor tamaño.

Si bien es verdad que a la hora de buscar una aplicación de control de producción nos podemos encontrar con aplicaciones como Odoo, perteneciente a Aures TI Consultores, S.L. la cual de la misma manera que en nuestro caso realizan una aplicación de producción teniendo en cuenta las necesidades del cliente al pie de la letra. Pero en nuestro caso, el directivo de la empresa no precisaba de un software tan especializado y profesional, sino que más bien precisaban de algo más funcional y preciso para cada uno de sus trabajadores en especial, y a un coste asumible.

Por tanto, este proyecto busca llenar este vacío en el mercado, ofreciendo una aplicación móvil que sea fácil de usar y accesible para empresas de cualquier tamaño, al tiempo que cubre todas las funcionalidades requeridas por la empresa en cuestión. Con esta aplicación, se pretende mejorar la eficiencia en la gestión de la producción en la fábrica textil, lo que redundará en un ahorro de costos y un aumento de la calidad en la producción.

1.3 Objetivos del proyecto

Los objetivos de este trabajo de final de grado son los siguientes:

- Desarrollar una aplicación móvil para el control de la producción en una fábrica textil utilizando Flutter.
- Desarrollar un servicio RESTful mediante Flask capaz de comunicar una aplicación móvil con un servidor de una empresa textil.
- Proporcionar información en tiempo real sobre los artículos pendientes de producción y el estado actual de la producción.
- Utilizar una base de datos de la empresa para almacenar y acceder a la información sobre la producción.
- Permitir a los gerentes de la fábrica tener una visión completa y actualizada del estado de la producción en todo momento.
- Optimizar y mejorar la eficiencia en la gestión de la producción en la fábrica textil mediante el uso de esta herramienta.

Estos objetivos buscan resolver las necesidades de la empresa y mejorar su capacidad de producción, al mismo tiempo que proporcionan una herramienta eficiente y práctica para la gestión de la producción.

1.4 Estructura del resto de la memoria

En cada uno de los puntos restantes, procederé a redactar lo siguiente:

2. **Análisis del problema**: En este apartado se profundiza en la definición del problema que se quiere solucionar con la aplicación, se analizan sus causas y consecuencias, se describen los posibles impactos que puede tener la solución y se realizan comparativas con soluciones ya existentes en el mercado.
3. **Solución del problema**: Este apartado se divide en diferentes subapartados donde se describe la arquitectura de la aplicación, se explican las tecnologías utilizadas y se detallan las diferentes capas de la aplicación. Además, se pueden incluir diagramas y esquemas que ayuden a entender mejor la solución propuesta.
4. **Tour por la aplicación:** En este apartado se realiza un recorrido por la aplicación, mostrando sus principales funcionalidades y características. Se pueden incluir capturas de pantalla o vídeos para ilustrar mejor las diferentes partes de la aplicación.

5. **Conclusiones y trabajos futuros:** En este apartado se realiza una reflexión sobre los resultados obtenidos con la aplicación, se evalúa si se han cumplido los objetivos marcados y se proponen posibles mejoras o trabajos futuros que se podrían llevar a cabo para seguir evolucionando la aplicación.
6. **Bibliografía**: Aquí se incluyen las fuentes bibliográficas y documentales utilizadas para el desarrollo del TFG, tanto libros, artículos científicos, sitios web, entre otros.
7. **Anexos**: Este apartado es opcional y se utiliza para incluir información adicional que puede complementar la memoria, como, por ejemplo, manuales técnicos, código fuente, encuestas, entre otros.

2. Análisis del problema

La empresa en cuestión a la que nos referiremos en lo que resta de memoria se trata de una pequeña empresa textil con 2 tiendas físicas en Alcoy y Albacete, que se dedica a la confección de prendas para el hogar; adquieren telas de un tercero para posteriormente acolchar o corta las telas para posteriormente realizar las colchas, cojines, edredones, etc.

En el ámbito productivo, con tal de poder entender el problema en cuestión que trata de solucionar la aplicación se realizará una descripción detallada de cómo se trabaja en la empresa.

Primeramente, la empresa dispone de un servidor SQL Server en el que se almacena toda la información sobre artículos y pedidos, a él acceden los trabajadores de oficina mediante una aplicación que ya les había sido diseñada con anterioridad y los trabajadores de la planta de fabricación acceden a los datos que les brinda la oficina mediante unas fotocopias que les son entregadas diariamente con la información de los productos que deben manufacturar esa jornada de trabajo. Es justo en ese punto donde la aplicación entra en acción, ya que se encargará de informar a los trabajadores que deben hacer en cada momento, a tiempo real.

Ya hablando sobre el proceso de producción, como ya se ha mencionado con anterioridad se adquieren las telas de un tercero para que estas puedan ser manufacturadas y convertidas en productos finales que serán directamente entregados al cliente o bien distribuidos a las tiendas físicas. Dependiendo del producto final que se desee fabricar estas telas se dirigirán a la sección de *Acolchado* o a la de *Corte*. Los productos para acolchar como pueden ser un boutí o una funda nórdica pasarán a la sección de *Acolchado* mientras que otros productos como cojines o conforts pasarán directamente a la sección de *Corte*, ya que no precisan acolchado. En la sección de *Acolchado* se trabaja de la siguiente manera, se obtienen las telas, y se obtiene el forro, se cortan a la medida indicada y posteriormente, con la unión ya lista para ser confeccionada se envía a las máquinas de confección. De la misma manera, en la sección de *Corte* se realiza el montado de las telas igualmente, y una vez está la tela montada, se envía de la misma manera a las máquinas de confección.

Cabe destacar una puntualización importante, en la sección de *Corte* puede darse el caso de que por cuestiones logísticas se demande una cantidad de producto y se realicen más, por ejemplo, puede ser que se pidan 3 cojines de cierto modelo, pero al realizar el corte resulten 4

cojines, esto no supondrá un problema ya que la aplicación se hará cargo de ello, todos los artículos de más se enviarán al almacén.

La confección trabaja bajo el mando del encargado de la sección de *Empaquetado*, de manera que para nuestro interés los productos se dirigirán directamente de las secciones de *Corte* y *Acolchado* a la sección de *Empaquetado*. La encargada de la sección de *Empaquetado* simplemente recibe los productos ya terminados, los pliega, los empaqueta, los etiqueta con su código correspondiente y se los envía al encargado de *Almacén*. Los envíos, con tal de aumentar la productividad se deben realizar por tandas, de manera que se envíe al almacén un conjunto de productos ya terminados. De esta manera, va escaneando tantos productos como la encargada desee y finalmente cuando se los vaya a entregar al encargado de *Almacén* realiza la confirmación en la aplicación. Como ya se ha puntualizado anteriormente, la gestión del stock es muy importante, ya que se pueden producir más artículos de los que se tengan pendientes por fabricar, es por eso por lo que cuando se le envía un producto al encargado de *Almacén* se le pueda enviar o bien para ser servido a un cliente o bien para que éste tenga que almacenarlo en el almacén. La aplicación controlará esta situación.

Finalmente, el encargado de *Almacén* se encarga de la distribución de todos los productos terminados o en stock a los clientes.

Recibe todos los clientes pendientes por servir mediante las hojas de envíos y realiza las cajas con los productos que le van llegando de la sección de *Empaquetado* para cuando ya tiene el envío completo o parcialmente completo realizar el albarán y preparar el envío. Por otro lado, el encargado de *Almacén*, en caso de necesitar un producto en concreto para terminar un envío debe ir preguntando por todas las secciones donde se encuentra ese producto en cuestión, la aplicación ayudará a hacer este trabajo de manera más eficiente pudiendo ir directamente a la sección en la que se encuentra el producto para pedir que éste se realice a la mayor brevedad posible.

A solicitud de la empresa, la aplicación incluye 4 perfiles distintos, a los que los trabajadores podrán acceder mediante un nombre de usuario y contraseña proporcionados por la empresa y almacenados en su base de datos. Por tanto, a partir de este momento, procederé a detallar los requisitos y funcionalidades específicas de cada uno de estos perfiles.

2.1 Acolchado

El usuario de acolchado dispondrá de las siguientes funcionalidades:

- **Selección de fecha límite**: El trabajador debe poder seleccionar una fecha límite de manera que se le muestren pedidos desde el pedido más antiguo hasta dicha fecha, que predeterminadamente es la actualidad.
- **Recepción de artículos pendientes**: la aplicación debe permitir la recepción de artículos pendientes de acolchado que no se encuentren en stock por orden de fecha de pedido.
- **Selección de medidas**: el trabajador de acolchado debe poder seleccionar la medida del artículo pendiente.

- **Selección de familia**: una vez seleccionada la medida, la aplicación debe mostrar las familias de productos disponibles en dicha medida para que el trabajador pueda seleccionar la correspondiente al artículo pendiente.
- **Selección de modelo**: después de seleccionar la familia, la aplicación debe mostrar los modelos disponibles para que el trabajador pueda seleccionar el correspondiente al artículo pendiente.
- **Selección de color**: una vez seleccionado el modelo, la aplicación debe mostrar los colores disponibles para que el trabajador pueda seleccionar el correspondiente al artículo pendiente.
- **Información adicional**: después de seleccionar la combinación de medida, familia, modelo y color, la pantalla debe mostrar la información adicional de las medidas restantes que necesita para completar dicha combinación.
- **Registro de acolchados realizados**: una vez que el trabajador ha acolchado el o los artículos seleccionados, la aplicación debe permitirle registrar la tarea realizada para que quede constancia en el sistema.

Finalmente, una vez el trabajador ha registrado el o los artículos volverá a la página de selección de medida.

2.2 Corte

El usuario de corte dispondrá de las siguientes funcionalidades:

- **Selección de fecha límite**: El trabajador debe poder seleccionar una fecha límite de manera que se le muestren pedidos desde el pedido más antiguo hasta dicha fecha, que predeterminadamente es la actualidad.
- **Selección de modelo**: El trabajador de corte debe poder seleccionar el modelo que se encuentra pendiente de corte.
- **Selección de color**: Una vez seleccionado el modelo, el trabajador de corte debe poder ver todos los colores disponibles para ese modelo y seleccionar uno.
- **Selección de familia**: Después de seleccionar el color, la aplicación debe mostrar las familias disponibles para ese modelo y ese color para que el trabajador de corte pueda seleccionar la que corresponda al pedido más viejo.
- **Selección de medidas**: Una vez seleccionada la familia, el trabajador de corte debe poder seleccionar las medidas a cortar, que pueden ser más de una.
- **Registro de cortes realizados**: una vez que el trabajador corte el o los artículos seleccionados, la aplicación debe permitirle registrar la tarea realizada para que quede constancia en el sistema.

Finalmente, una vez el trabajador ha registrado el o los artículos volverá a la página de selección de medida.

2.3 Corte-Luís

Bajo petición de uno de los trabajadores, realizamos un perfil a su medida, una variable del perfil de corte, que tiene las siguientes funcionalidades:

- **Selección de fecha límite**: El trabajador debe poder seleccionar una fecha límite de manera que se le muestren pedidos desde el pedido más antiguo hasta dicha fecha, que predeterminadamente es la actualidad.
- **Selección de línea de producción**: El trabajador de corte debe poder seleccionar la línea de producción de un producto que se encuentre pendiente de corte.
- **Selección de modelo**: Una vez seleccionado la línea, el trabajador de corte debe poder ver todos los modelos disponibles para esa línea y seleccionar uno de ellos.
- **Selección de color**: Una vez seleccionado el modelo, el trabajador de corte debe poder ver todos los colores disponibles para ese modelo y seleccionar uno.
- **Selección de familia**: Después de seleccionar el color, la aplicación debe mostrar las familias disponibles para ese modelo y ese color para que el trabajador de corte pueda seleccionar la que corresponda al pedido más viejo.
- **Selección de medidas**: Una vez seleccionada la familia, el trabajador de corte debe poder seleccionar las medidas a cortar, que pueden ser más de una.
- **Registro de cortes realizados**: una vez que el trabajador corte el o los artículos seleccionados, la aplicación debe permitirle registrar la tarea realizada para que quede constancia en el sistema.

Finalmente, una vez el trabajador ha registrado el o los artículos volverá a la página de selección de línea.

2.4 Empaquetado

El usuario de empaquetado dispondrá de las siguientes funcionalidades:

- **Recepción de artículo terminado**: La aplicación deberá permitir al trabajador de empaquetado recibir un artículo que haya sido terminado en el proceso de producción.
- **Escaneo de códigos de barras**: La aplicación deberá contar con un lector de códigos de barras integrado para permitir al trabajador de empaquetado escanear los códigos de los productos a empaquetar.
- **Selección y deselección de artículos**: La aplicación deberá permitir al trabajador de empaquetado seleccionar y deseleccionar artículos para ser empaquetados en caso de algún error.
- **Transferencia al almacén**: La aplicación deberá permitir al trabajador de empaquetado transferir los artículos empaquetados al almacén una vez que haya completado su empaquetado y esté listo para su almacenamiento.

2.5 Almacén

El encargado del almacén dispondrá de las siguientes funcionalidades:

- **Selección de pedidos**: El trabajador podrá seleccionar los clientes que esperan un pedido, ordenados por fecha de pedido.
- **Información de artículos**: Una vez seleccionado el pedido, se le mostrarán al trabajador los artículos terminados y los pendientes para completar el pedido.

- **Búsqueda de productos**: El encargado del almacén podrá buscar uno o más productos, y la aplicación le mostrará los clientes que precisen de estos artículos.
- **Búsqueda de clientes por productos**: el encargado de almacén podrá buscar clientes que precisen de uno o más productos en específicos.
- **Visualización de productos en otras secciones**: en caso de necesitar rastrear un artículo, el encargado de almacén podrá visualizar los productos que se encuentran en las otras secciones de la empresa (acolchado, corte, empaquetado).
- **Gestión de stock**: El encargado del almacén será responsable de mantener el stock de la empresa actualizado, así que podrá añadir nuevos productos y actualizar los existentes.
- **Gestión de pedidos**: El encargado del almacén será responsable de gestionar los pedidos y de asegurarse de que se entreguen a los clientes en el plazo previsto.

Finalmente, una vez el trabajador ha registrado el envío se actualizan los artículos y los clientes para así poder continuar con el siguiente envío.

Por otro lado, la empresa también requiere los siguientes requisitos no funcionales:

- **Usabilidad:** La aplicación deberá de poder usarse en el entorno de trabajo, siendo así fácil de usar.
- **Perfil sencillo para acabados:** El perfil de acabados deberá tener un funcionamiento básico que sea fácil de explicar a una persona mayor.
- **Base de datos SQL:** La aplicación deberá trabajar con el servidor de la empresa sin poder modificarlo, tan solo consultarlo.
- **Autentificación:** El empleado se deberá autenticar con su perfil con tal de poder entrar a la aplicación.
- **Rendimiento:** El rendimiento de la aplicación debe garantizar que es capaz de mantener el ritmo de la producción, de manera que no ralentice al trabajador.
- **Seguridad:** Toda la información con la que trabaja la empresa no debe ser visible desde el exterior de la empresa
- **Confidencialidad:** La información de los clientes solo podrá ser visible por el empleado de *Almacén*

En la Figura 1, se puede observar los diferentes pasos que se siguen para, desde el servidor principal de la empresa se llegue al artículo finalizado y listo para ser distribuido por el encargado del almacén.

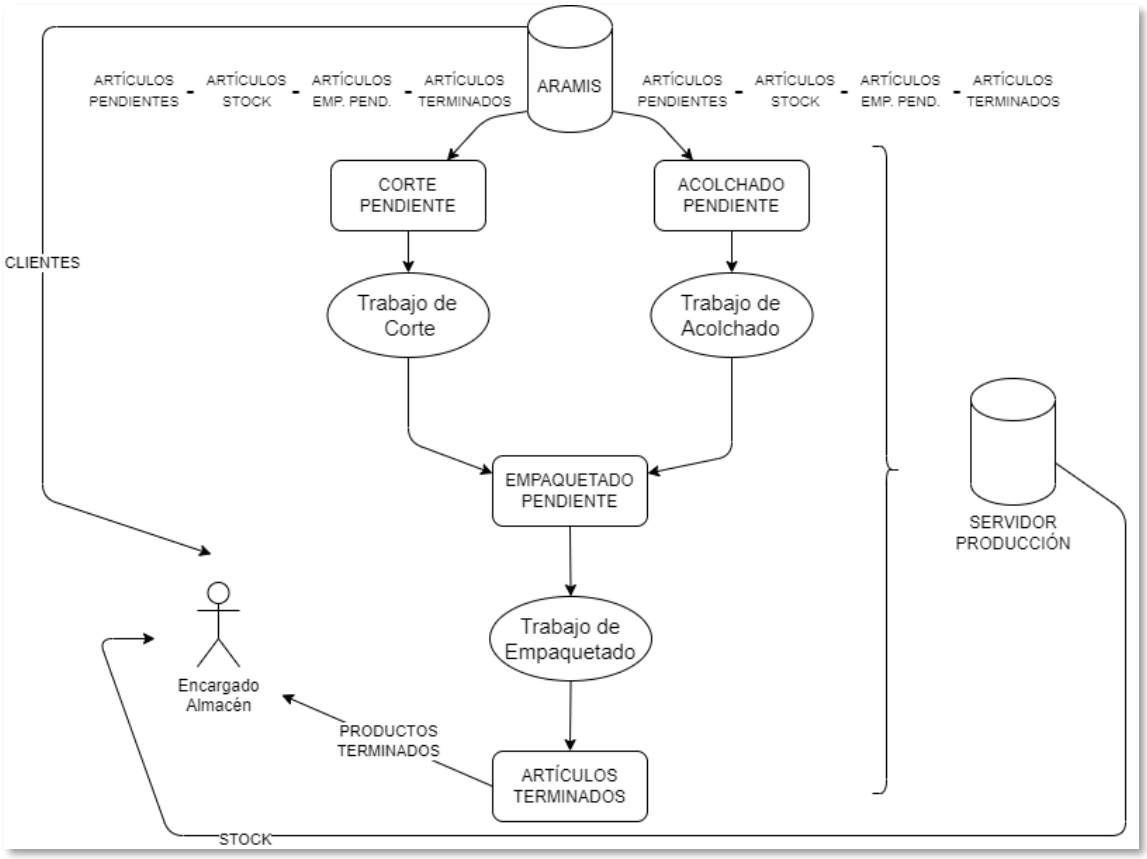

Figura 1: Diagrama de flujo sobre el funcionamiento de la aplicación

3. Solución del problema

A continuación, se van a detallar las decisiones y los procesos seguidos para dar solución al problema planteado. Se presentará la solución propuesta para satisfacer los requisitos establecidos en el apartado 2. Es por ello por lo que se describirán tanto las tecnologías utilizadas como la arquitectura de la aplicación o las distintas capas de ésta. Además, se explicarán las decisiones tomadas en cuanto a la organización de los diferentes perfiles dentro de la aplicación, con el fin de garantizar una experiencia de usuario óptima y un funcionamiento eficiente del sistema.

Como ya se mencionó previamente, esta aplicación contiene en su interior diferentes perfiles para cada trabajador en particular. Por lo tanto, en esta sección se describirá tanto los detalles genéricos de la aplicación como cada uno de los perfiles de forma detallada por separado.

3.1 Arquitectura de la aplicación

Así pues, la aplicación se ha diseñado con una estructura de tres capas: la capa de presentación de datos, la capa de lógica de negocio y la de acceso a datos.

La capa de presentación de datos es la interfaz gráfica de la aplicación, donde el usuario utilizará la pantalla táctil de su móvil o tableta para poder interactuar con ella. En esta interfaz el usuario encuentra botones de diferentes tipos, selecciones de fechas e incluso un lector de códigos de barras para ciertos perfiles. Con tal de lograrlo se ha desarrollado mediante Flutter.

En la capa de lógica de negocio se utiliza Python, en esta capa es en la que se encuentra la funcionalidad principal de la aplicación, en ella se procesan los datos que se reciben de la capa de interfaz de usuario y con los datos obtenidos de la capa de acceso a datos le devuelve una respuesta de nuevo al usuario.

Además, también tenemos por otro lado el servicio web RESTful implementado en Flask que se encarga de exponer la funcionalidad de la capa lógica de negocio a través de una API web. Actúa como la interfaz entre la capa de presentación y la capa de lógica de negocio.

Finalmente, en la capa de acceso de datos es la que se encarga de interactuar con la base de datos de la empresa en Microsoft SQL (*ARAMIS*), así como con una base de datos MySQL auxiliar que ha sido desplegada para el correcto funcionamiento de la aplicación. El uso de la base de datos MySQL se debe a, como ya se ha explicado en los requisitos no funcionales del apartado 2, que la empresa no nos permite realizar modificaciones en su base de datos, por lo que tan solo nos permite realizar consultas, eso nos obliga a diseñar una base de datos auxiliar para mantener toda la información que precisa la aplicación para poder funcionar (los productos que se encuentran en cada una de las secciones de la empresa, los artículos finalizados, el stock,...). Esta capa también trabaja proporcionando los datos necesarios para la capa de lógica de negocio y almacenando los resultados de las operaciones realizadas en esta capa.

Asi pues, como ya se ha explicado anteriomente, el cliente trabajará sobre la capa de presentación, que responderá ante él comunicandose con el backend. Con tal de poder cumplirse esa comunicación se utiliza un servicio web RESTful Flask, implementado mediante Python, y trabajan conjuntamente de manera que la aplicación emite peticiones http al servidor web mediante un paquete disponible en Flutter. Una vez la capa de presentación de datos envia la petición http la capa de lógica de negocio realiza sus calculos internos y sus comunicaciones con la capa de acceso a datos para así finalmente brindar la información necesaria. Por otro lado, la comunicación entre el la capa de acceso a datos y la capa de logica de negocio es un tanto especial, ya que contará de dos conexiones, una a la base de datos de MS SQL y otra para la conexión de MySQL.

Tanto en la comunicación entre la capa de presentación y la capa de lógica de negocio como en la comunicación la capa de lógica de negocio y la capa de acceso a datos existen mecanismos de Autentificación y Seguridad. Por un lado, con respecto a la comunicación entre la capa de presentación y la capa de lógica de negocio, tan solo los dispositivos conectados a la red de la empresa, que está protegida con seguridad WPA2-AES, podran comunicarse con el servicio web RESTful. Por otro lado, la comunicación entre la capa de lógica de negocio y la capa de acceso a datos, en ambas conexiones se utiliza autentificación con usuario y contraseña además de securiza las conexiones TCP/IP mediante SSL.

En la Figura 2 presentada a continuación se puede observar la arquitectura de la aplicación de una manera más fácil y sencilla, así como observar de manera resumida las interacciones entre los diferentes componentes de ésta.

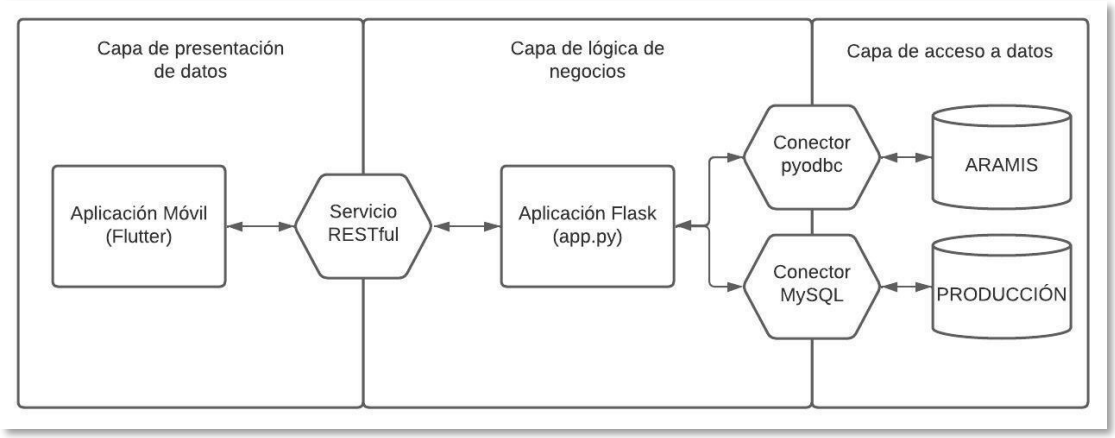

Figura 2: *Diagrama de bloques sencillo que muestra las 3 capas de la aplicación*

3.2 Tecnologías utilizadas

Con tal de poder llevar a cabo la aplicación se ha recurrido a las siguientes tecnologías:

- Flutter mediante Android Studio, con tal de desarrollar la capa de presentación se ha utilizado Flutter dado que ya tenía conocimientos previos cosa que me ha servido a la hora de decidir que tecnología utilizar. Se ha decidido trabajar con esta tecnología por delante de otra más clásica como Kotlin dado que con Flutter tenía posibilidad de acceder a complementos más novedosos dado su creciente popularidad entre la comunidad de desarrolladores.
- Python con mediante Visual Studio Code, a fin de implementar la capa lógica y la capa de acceso a datos se ha utilizado Python de manera que importando librerías conseguimos el objetivo que precisábamos. Por el mismo motivo que anteriormente, se ha elegido Python por delante de otros lenguajes de programación como pueden ser Java o C++ por La enorme comunidad que se está formando en torno a Python, que está mejorando todos los aspectos del lenguaje y por la comodidad que me aporta a la hora de trabajar con él, una comodidad que no comporta a cambio una pérdida de rendimiento, por lo tanto, es perfecto para mis necesidades.
- Flask, con tal de comunicar la capa de interfaz de usuario con la capa de aplicación crearemos una aplicación web con Flask en Python de manera que se comuniquen mediante HTML. Como alternativa a Flask como servicio web en Python podemos encontrarnos con Django, que, aunque más antiguo, no es capaz de ofrecernos la ligereza y simplicidad que es capaz de ofrecernos Flask, y es que Flask desde su construcción fue concebida con esos conceptos de simplicidad y escalabilidad en mente. Al ser una tecnología ligera, fácil de adoptar, perfectamente documentada y con una popularidad reconocida me ha parecido la mejor opción con tal de desarrollar mi servicio web RESTful.
- SQL Server Management Studio, la empresa trabaja con un SQL Server y utilizaremos esta herramienta con tal de gestionar el servidor SQL.

- Conector de Python con SQL Server, la capa de aplicación se comunicará con el SQL Server gracias al conector Pyton SQL driver - pyODBC.
- MySQL Workbench, la empresa no deseaba que se modificase las tablas de la base de datos de MS SQL Server, es por eso por lo que necesitamos un servidor MySQL server y MySQL Workbench con tal de poder mantener el control de la producción incluyendo los albaranes y el control de stock.
- Conector de Python con MySQL, la capa de aplicación se comunicará con el MySQL Server gracias al conector MySQL Connector/Python.
- Diferentes componentes de Flutter, con tal de realizar una aplicación completa no nos sirve con la versión básica de Flutter es por ello por lo que necesitamos los siguientes componentes:
- *cupertino_icons*: Utilizamos este componente para utilizar unos iconos mejores a los que tenemos de manera básica.
- *multi_select_flutter*: Es un paquete para crear widgets de selección múltiple de varias maneras.
- *http*: Utilizaremos este paquete para poder comunicar la aplicación móvil con la aplicación web de Flask que está desplegada en el servidor de la empresa.
- *flutter_barcode_scanner*: Este paquete se trata de un escáner de código de barras que utilizaremos en el perfil de empaquetado.
- GitLab, utilizaremos GitLab para poder mantener un control de las versiones de la aplicación.

3.3 Capa de presentación de datos

En este apartado vamos a diferenciar entre los diferentes perfiles de los trabajadores ya que así se puede realizar una definición más detallada de cada perfil y sus funcionalidades específicas. A pesar de esto, cabe señalar que de manera general la interfaz del usuario está diseñada para ser muy sencilla y moderna, de manera que un trabajador no nativo digital sea capaz de trabajar con ella sin que el desconocimiento tecnológico suponga una barrera. En resumen, la interfaz de usuario de la aplicación se ha diseñado con el objetivo de proporcionar una experiencia fácil de usar e intuitiva para los usuarios, utilizando un diseño moderno y minimalista, iconos y etiquetas descriptivas, además de en algunos casos diseñar en particular la aplicación bajo sus gustos particulares.

También cabe destacar que la aplicación está enfocada tanto a ser utilizadas por tabletas como por teléfonos móviles es por ello por lo que tiene la capacidad de adaptarse a diferentes tamaños de pantalla.

3.3.1 Acolchado

En el perfil de acolchado, el objetivo de la aplicación es que el empleado elija un producto para acolchar teniendo en cuenta que se deben acolchar los pedidos más antiguos primero. Para ello la aplicación muestra un selector de fecha en la parte superior que permite

seleccionar hasta que fecha de pedidos deseas que se muestren los productos. De manera predeterminada, la fecha seleccionada será la actual, y posteriormente la aplicación tras obtener los datos de la capa de aplicación mediante la utilización de un *Future*[1] mostrará una *GridView*[2] con botones que al pulsar dirigirán a la siguiente pantalla. Esto será un procedimiento generalizado en el resto de la aplicación para la mayoría de los perfiles.

En el caso del perfil de acolchado en esta *GridView* se mostrarán primeramente las medidas, una vez seleccionada una medida, el empleado deberá seleccionar una familia de productos. Posteriormente, se deberá seleccionar un modelo de producto y finalmente un color.

En el momento en el que el empleado selecciona un color, un *Drawer*[3] se podrá desplegar con tal de proporcionarle información al usuario sobre las medidas restantes para esta selección de familia, modelo y color.

La interfaz de usuario también proporcionará un botón para marcar el artículo como completado una vez el empleado tenga claro el artículo que desea acolchar.

3.3.2 Corte

Para el perfil de corte, tenemos el mismo objetivo que en el perfil anterior, es por eso por lo que utilizaremos la misma estructura y diseño. Es por eso por lo que tan solo se detallarán aquellas características distintivas que encontramos entre ambos perfiles.

En esta sección, en primer lugar, se mostrarán las diferentes líneas de producción disponibles para que el trabajador pueda seleccionar la que corresponda al pedido más antiguo. A continuación, se presentarán las familias de productos disponibles para esa línea. Una vez elegida la familia, se desplegarán los modelos de producto disponibles para seleccionar el que el trabajador desee. Por último, se mostrarán las medidas disponibles para ese modelo en particular.

Se le mostrará al trabajador en un *Drawer*, todas las medidas disponibles para las características seleccionadas y las cantidades que se necesitan para cada uno de esos artículos.

---

[1] *Future: El resultado de una computación asíncrona. En lugar de bloquear toda la computación hasta que el resultado esté disponible, la computación asíncrona devuelve inmediatamente un "se completará" hasta que se obtiene el resultado.*

[2] *Grid View: Una matriz de widgets 2D desplazable.*

[3] *Drawer: Un panel Material Design que se desliza horizontalmente desde el borde de un "Scaffold" para mostrar enlaces de navegación en una aplicación.*

El trabajador en este caso podrá confirmar de dos maneras diferentes, como en el perfil anterior, o bien de manera que si se da el caso de que haya más medidas a seleccionar con este modelo y color se le informará al trabajador con el pop-up de confirmación de que consulte el *Drawer.* El trabajador también podrá confirmar el artículo mediante el *Drawer,* donde en este caso se le redirige al trabajador de nuevo a la pantalla final con la nueva medida seleccionada.

3.3.3 Corte-Luís

Esta sección ha sido diseñada como una modificación de la sección de corte para un trabajador en concreto que la deseaba de esta manera.

En este caso, tan solo hay una pequeña variación interna a nivel de aplicación, y es por eso por lo que a pesar de que sean diferentes internamente, a nivel de interfaz son idénticas.

3.3.4 Empaquetado

En el perfil de empaquetado el diseño es totalmente diferente, ya que el objetivo de este perfil es tan solo escanear los códigos de barras de los productos empaquetados para ser enviados a almacén.

El interfaz de este perfil es básico y sencillo ya que tan solo contiene cuarto elementos: Un selector de fechas como en las secciones anteriores, un escáner de código de barras, una lista de los artículos escaneados y un botón con tal de poder confirmar la lista de los artículos escaneados.

El empleado tendrá la capacidad de eliminar un producto de la lista tan solo tocando sobre él, para poder rectificar en caso de error.

3.3.5 Almacén

Para el perfil de almacén, el paradigma es totalmente diferente ya que necesita un comportamiento más complejo. En este caso, el objetivo del empleado de almacén es el de servir los pedidos a los clientes. Para eso la aplicación le brindará varias herramientas.

Por un lado, el trabajador deberá de poder seleccionar entre todos los clientes, ordenados por fecha de pedido y que la aplicación le muestre todos los artículos disponibles de servir para poder realizar el pedido. Al trabajador también se le informará de los artículos pendientes de producir para poder completar el pedido de dicho/s cliente/s.

Por otro lado, el trabajador podrá seleccionar entre todos los artículos terminados, ordenados por fecha de pedido y una vez seleccionado uno o varios artículos la aplicación le mostrará los clientes que necesitan este conjunto de artículos.

Finalmente, con tal de poder mantener un control de los artículos, el trabajador también será capaz de observar en la sección en la que se encuentra cada uno de los artículos. Además,

el encargado de almacén también será el responsable del stock de la empresa es por eso por lo que se le deben brindar las herramientas para que así pueda ser.

En la Figura 3, se puede observar cuál es la distribución de ficheros que hace posible esta capa, dentro de la carpeta *screens* se organizan las diferentes carpetas de secciones donde se encuentran los ficheros .dart que las componen. Por otro lado, existe el fichero main.dart que inicia la aplicación y el fichero login_page.dart que es el encargado de realizar la autentificación del empleado para redirigirlo a la sección correcta.

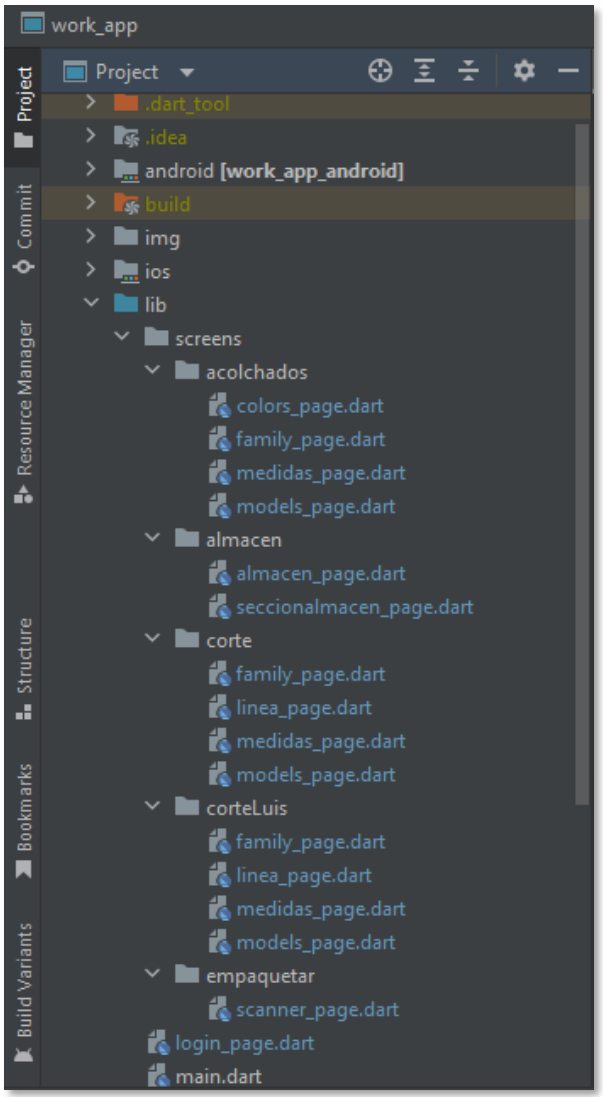

Figura 3: Captura de pantalla con la distribución de los archivos .dart

Todos los ficheros siguen más o menos un mismo patrón generalizado, de manera que, para que se pueda entender de mejor manera el comportamiento de la aplicación a continuación se realizará un pequeño análisis de cuáles son estos pasos que sigue la aplicación a la hora de mostrar los datos al usuario, esperar su respuesta y trabajar en consecuencia para que así sea mucho más fácil de comprender como se ha trabajado con Flutter.

De manera general, cada vez que el usuario navega hasta la siguiente página ésta principalmente trabaja con clases y funciones.

Así pues, cuando se navega a una pantalla inicialmente se carga el widget "*build"* que tienen todos los ficheros dart y que se encarga de realizar un "*return"* de un *"child"*, para que quede más claro, al cargar la pantalla de la aplicación dart internamente devuelve un componente de la aplicación, en el que están insertados los demás componentes.

En nuestro caso, habrá un gran número de veces, que dart habrá de esperar a obtener resultados del servidor web, de manera que mientras no los recibe debe mostrar un círculo de carga. Una vez obtenidos los resultados del contenido asíncrono se los mostrará al cliente para que éste pueda seguir usando la aplicación.

Cuando dart ya nos presenta el componente con todos los demás componentes cargados interiormente, se espera a recibir el feedback del usuario para actuar en consecuencia, una vez el usuario haya decidido interactuar con alguno de los componentes la aplicación responde, en nuestro caso, la mayoría de veces el usuario haya escogido alguna característica de un artículo o bien un cliente o un artículo, de manera que, una vez el cliente se ha decidido o bien se le redirige a la siguiente página o bien esperamos a una confirmación con tal de actual, así pues cuando finalmente el usuario ha escogido y/o confirmado pasaremos a la siguiente "screen".

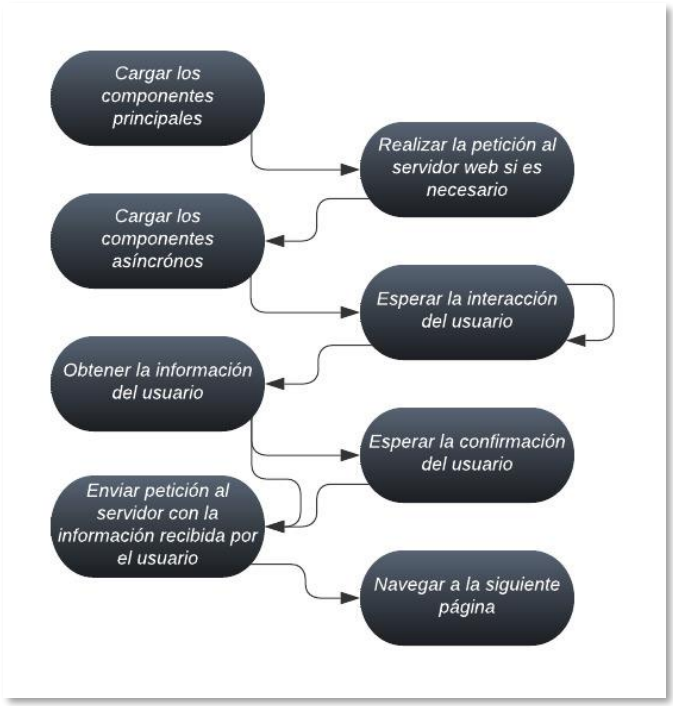

Figura 4: Diagrama de flujo explicativo sobre cómo actúan los ficheros .dart

Una vez explicado los elementos, se mostrarán unos mockups para que así ilustren mejor todo lo que ha sido explicado en esta sección. Se podrán ver mockups de dos tipos, los realizados para las secciones de producción que posteriormente han sido levemente modificados bajo las peticiones de cada trabajador en concreto y los realizados para la sección del almacén. Se debe puntualizar que a pesar de que la sección que se muestra en los mockups es la sección de *Acabados* ésta fue suprimida por parte del director de la empresa que decidió que no debían utilizar la aplicación.

Figura 5: Mockup de la pantalla de inicio de la aplicación

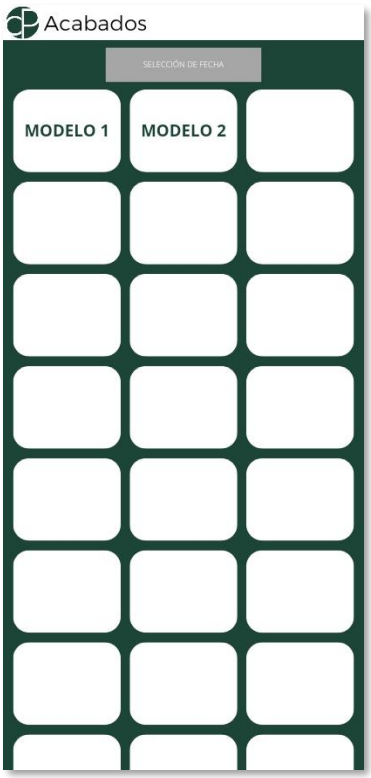

Figura 6: Mockup de la producción donde se muestra el GridView de selección de modelo

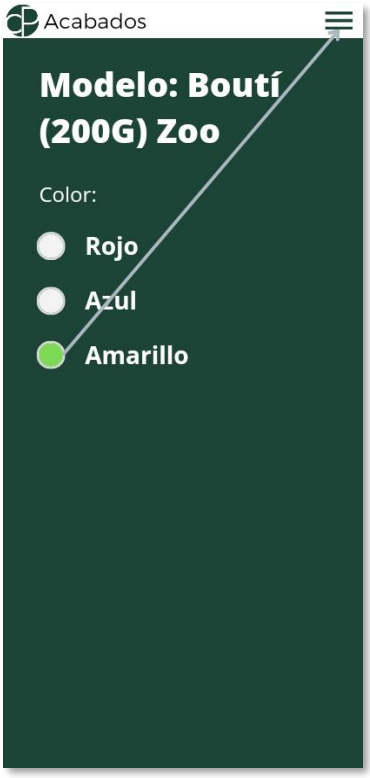

Figura 7: Mockup de la producción donde se muestra selector de color y la interacción del empleado

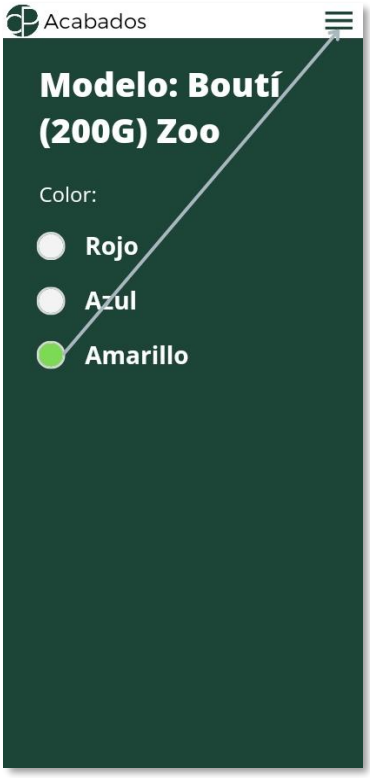

Figura 8: Mockup de la producción donde se muestra el Drawer con la selección de medidas y la confirmación

Figura 9: Mockup del almacén donde se muestra la pantalla principal a la cual vuelve tras cada envío

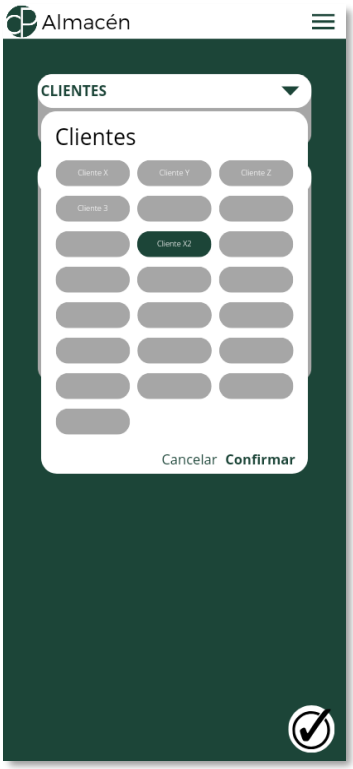

Figura 10: Mockup del almacén donde se muestra el MultiSelectDialogField de selección clientes

Figura 11: Mockup del almacén donde se muestra el MultiSelectDialogField de selección productos

Figura 12: Mockup del almacén donde se muestra el Drawer para visualizar las secciones a controlar

Figura 13: Mockup del almacén donde se muestra una de las secciones controladas visualizando que se está haciendo

Aunque todos estos bocetos ayuden a hacernos una idea de cómo se visualiza la aplicación, ésta se ha de codificar, y así pues a continuación se mostrará un fragmento de código perteneciente a la sección del almacén, se trata de un botón encargado de confirmar el envío y realizar el albarán. Éste se trata de un Future, lo cual significa que es una sección del código asíncrona, por lo tanto, mientras se realiza la petición http al servidor web la aplicación sigue funcionando y nos muestra el resultado una vez lo reciba.

```
Future<String> albaran(List<dynamic> selectedClients, List<dynamic>
selectedProducts) async {
    if (selectedProducts.isEmpty || selectedClients.isEmpty) {
      return Future.error("...");
    } else {
      var selection = {
        "productos": selectedProducts,
        "clientes": selectedClients
      };
      http.Response response = await http.post(
          Uri.parse("http://192.168.0.10:8088/almacen/albaran"),
          headers: {'Content-Type': 'application/json'},
          body: json.encode(selection));
      if (response.statusCode == 200) {
        var result = jsonDecode(utf8.decode(response.bodyBytes));
        return result.toString();
      }else{
        return Future.error('fallo en la respuesta del servidor');
      }
    }
  }
child: FloatingActionButton(
                  onPressed: () async {
                    var alb = await albaran(selectedClients,
selectedProducts); showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return AlertDialog(
                            title: const Text('Pedidos albaranados'),
                            content: Text(alb.toString()),
                            actions: <Widget>[
                              TextButton(
                                style: TextButton.styleFrom(
                                  textStyle:
                                ),
                                child: const Text('Okay'),
                                onPressed: () {setState(() {
                                  selectedClients = [];
                                  selectedProducts = [];
                                  Navigator.pop(context);
                                });},
                              ),
                            ],
                          );
                        },
                      );
                  },
                  backgroundColor: Colors.white,
                  child: const Icon(Icons.check,
                      color: Color.fromRGBO(52, 103, 101, 1)),
                ),
```

Figura 14: Fragmento de código perteneciente a la capa de presentación de datos

3.4 Capa de lógica de negocio

En este apartado vamos a analizar la capa lógica de negocio, la encargada de procesar todos los datos introducidos por el usuario y aplicar las operaciones necesarias sobre ellos. En el caso de nuestra aplicación, podemos diferenciar entre tres tipos de sección, la de producción donde se encuentran las secciones de corte y la de acolchado, la sección de empaquetado y la de almacén.

Para las secciones de producción, la capa de aplicación tan solo debe realizar un trabajo y es ir mostrándole al trabajador las características demandadas en cada momento para que así éste pueda seleccionar el artículo deseado y una vez seleccionado permitirle la confirmación del producto para enviarlo a la siguiente sección. Esto se consigue de manera que el interfaz de usuario y la capa de aplicación se comunican realizando peticiones HTML a la aplicación y ésta obtiene los datos y le responde en el cuerpo de la página HTML de donde la interfaz obtiene los datos y se los muestra al usuario.

Mientras tanto, para la sección de empaquetado en este caso es más simple, ya que simplemente tiene dos funciones. Por un lado, la aplicación debe obtener la lista de artículos escaneados para ir catalogándolos como artículos terminados, de manera que se recorre la lista de los artículos y se va añadiendo el artículo o modificando el valor cantidad de éste en la tabla de artículos terminados. Por otro lado, la aplicación obtiene el código que el empleado escanea para obtener la información del artículo para que así el empleado sea capaz de reconocer el artículo de manera fácil.

Finalmente, para la sección de almacén vamos a diferenciar dos labores de la aplicación, la de mostrar la información respecto a los pedidos y mostrar la información respecto a loa artículos en producción. Con relación a los pedidos, la aplicación primeramente carga y envía todos los artículos terminados y los clientes pendientes de servir al usuario ambos por orden de antigüedad de pedido, y posteriormente, en caso de que se seleccionen uno o más productos o clientes, la aplicación deberá cargar y enviar al usuario los productos o clientes correspondientes. Una vez seleccionados producto/s y cliente/s la aplicación debe confirmar el envío. Con relación a los artículos en producción, la aplicación le debe ofrecer al usuario la información sobre todos los artículos pendientes de la sección solicitada.

Como se ha realizado anteriormente en la sección 3.3, a continuación, se procede a mostrar un fragmento de código del archivo "*app.py*" que contiene la aplicación que será desplegada por Flask y la que será posteriormente consultada por la capa de presentación de datos para ofrecerle la información al cliente. Más concretamente este fragmento pertenece a la sección de acolchado que es bastante delicado, eso es porque es el fragmento de código que se encarga de mostrarle al trabajador las medidas disponibles para acolchar y es crítico porque no se deben mostrar artículos ya acolchados previamente y es por eso por lo que debemos mantener un control exhaustivo de los productos que nos muestra.

Primeramente, podemos observar el encabezado del servicio web, la dirección http mediante la que, el servicio Flask encontrará este fragmento de código para que, con los argumentos brindados por el usuario, éste pueda recibir la respuesta deseada y no cualquier otra.

```
#ACOLCHADO
@app.route('/acolchado/medidas')
def get_acomedidas():
    args = request.args
    fecha = args.get('fecha')
    cursor = connARAMIS.cursor()
    cursor.execute(f"SELECT artcod, COUNT(*) as cantidad FROM vpel INNER JOIN
vpe ON...")
    pedidos = []
    rows = cursor.fetchall()
    for row in rows:
        pedidos.append([x for x in row])

    #ELIMINAR DE LOS PRODUCTOS A MOSTRAR QUE SE ENCUENTRE EN STOCK

    stockCursor = connPROD.cursor()
    stockCursor.execute("SELECT artcod, cantidad FROM stock_inv;")
    stock = []
    myrows = stockCursor.fetchall()
    for row in myrows:
        stock.append([x for x in row])

    i=0
    while (i < len(pedidos)):
        elemento = pedidos[i][0]
        cantidad = pedidos[i][1]
        encontrado = False
        j = 0
        while encontrado == False and j < len(stock):
            pedidoStock = stock[j]
            if pedidoStock[0] == elemento:
                nuevaCantidad = cantidad - pedidoStock[1]
                if nuevaCantidad > 0:
                    pedidos[i][1] = nuevaCantidad
                elif nuevaCantidad <= 0:
                    pedidos.pop(i)
                    i-=1
                encontrado = True
            j+=1
        i+=1

    # REPETIMOS LO ANTERIOR CON LOS PRODUCTOS TERMINADOS, LOS EMPAQUETADOS Y
LOS ALBARANADOS

    prodCursor = connPROD.cursor()
    prodCursor.execute(f"TRUNCATE TABLE produccion.acolchado_pendiente;")
    for pedido in pedidos:
        ped = tuple(pedido)
        prodCursor.execute(f"INSERT INTO produccion.acolchado_pendiente VALUES
{ped};")
    prodCursor.close()
    connPROD.commit()

    artcods = []
    for cod in pedidos:
        artcods.append(cod[0])

    global artcods_acolchado
    artcods_acolchado = artcods.copy()

    if len(artcods) == 1:
        artcod = artcods[0]
        artcods = f"('{artcod}')"
    else:
        artcods = tuple(artcods)
        if artcods == ():
            artcods = "('')"
```

```
    cursor.execute(f"Obtener las medidas ordenadas por fecha de pedido")

    medidas = []
    rows = cursor.fetchall()
    for row in rows:
        medidas.append(row[0])
    medidas_distinct = []
    for medida in medidas:
        if medida not in medidas_distinct:
            medidas_distinct.append(medida)
    return medidas_distinct
```

Figura 15: Fragmento de código perteneciente a la capa de lógica de negocio

Como habéis podido observar en el código, Flask accede a este fragmento de código con un argumento brindado por el usuario, lo obtenemos y se obtienen todos los productos que deben ser acolchados hasta dicha fecha. Posteriormente se han de retirar todos aquellos que, por cuestión de falta de actualización de las bases de datos de la empresa, se mantienen hasta varias semanas en la base de datos productos ya enviados, es por eso por lo que del total de artículos que recibimos vamos retirando todos aquellos que no deseamos. Una vez tenemos todos los códigos con las cantidades necesarias para acolchar procedemos a guardarlo en una variable global para que así sea usada por el resto de las funciones que lo necesiten. Una vez tenemos el código de los productos deseados, con tal de poderle mostrar al trabajador las medidas disponibles de manera única y ordenados por fecha de pedido más antiguo realizamos la consulta de las medidas de los productos con los códigos que tenemos en nuestra lista *"artcods"* de manera que posteriormente eliminamos las repetidas y le devolvemos al trabajador una lista con las medidas sin repeticiones y ordenadas.

Como ya se ha mencionado anteriormente, Flask tiene todas las rutas de la aplicación guardadas para posteriormente poder dirigirse al código y trabajar con los datos que le pasa el cliente como argumentos y así poder computar una respuesta. A continuación, se mostrará una tabla con todas las rutas que utilizamos en la aplicación.

@app.route('/')	Método predeterminado que da la bienvenida a la API.
@app.route('/fecha')	Método que obtiene la fecha del pedido más antiguo.
@app.route('/login/user')	Método que comprueba que el usuario existe en la base de datos.
@app.route('/login/password')	Método que comprueba que la contraseña del usuario existe en la base de datos y es correcta.
@app.route('/login/seccion')	Método que obtiene la sección del usuario una vez éste se ha autentificado.
@app.route('/[sección]/medidas')	Método generalizado para cada sección que obtiene las medidas de los productos a trabajar.
@app.route('/[sección]/familia')	Método generalizado para cada sección que obtiene las familias de los productos a trabajar.
@app.route('/[sección]/modelos')	Método generalizado para cada sección que obtiene los modelos de los productos a trabajar.

@app.route('/[sección]/color')	Método generalizado para cada sección que obtiene los colores de los productos a trabajar.
@app.route('/acolchado/infomedidas')	Método de la sección de acolchar que informa sobre las demás medidas disponibles para acolchar de ese mismo modelo.
@app.route('/corte/linea')	Método de la sección de corte que obtiene las líneas de producción de los productos que se cortan.
@app.route('/corteLuis/linea')	Método de la sección de corte que obtiene las líneas de producción de los productos que se cortan, aunque con unas diferenciaciones respecto al corte predeterminado.
@app.route("/[sección]/finalizar")	Método generalizado para cada sección que indica que el producto ha sido finalizado y que pasa a enviarlo a la siguiente sección de la producción.
@app.route("/empaquetado/finalizar", methods=['POST'])	Método de la sección de empaquetado que indica que el producto ha sido finalizado y que pasa a enviarlo a la siguiente sección de la producción. Tiene la peculiaridad que se le envían los parámetros mediante un json lo que nos hace indicarle al servidor que esta ruta requiere del método *"POST".*
@app.route('/almacen/clientes')	Método de la sección de almacén que nos ofrece los clientes pendientes de servir.
@app.route('/almacen/articulos')	Método de la sección de almacén que nos ofrece los artículos terminados pendientes de servir.
@app.route('/almacen/clienteSelect', methods=['POST'])	Método de la sección de almacén que nos ofrece los clientes que necesiten el conjunto de productos que se envía en la petición http.
@app.route('/almacen/articuloSelect', methods=['POST'])	Método de la sección de almacén que nos ofrece los productos que necesiten el conjunto de clientes que se envía en la petición http.
@app.route('/almacen/[sección]')	Método de la sección de almacén que muestra los productos pendientes de cada sección, en este caso generalizadas como [sección].
@app.route('/almacen/albaran', methods=['POST'])	Método de la sección de almacén con la que se realiza el albarán de los productos y clientes seleccionados que se envían con la petición http.

Figura 16: Tabla con todas las rutas publicadas en el servicio RESTful con una breve explicación sobre su comportamiento

Finalmente, con tal de que sea más fácil inferir cuál es el comportamiento de la aplicación y como todos los métodos que componen la aplicación desplegada en Flask siguen un comportamiento generalizable se ha realizado un diagrama que se mostrará a continuación.

Figura 17: Diagrama de flujo explicativo sobre cómo actúan los métodos de la aplicación desplegada en Flask (app.py)

3.5 Capa de acceso a datos

La capa de acceso a datos es una de las más importantes en cualquier aplicación, ya que es la responsable de la gestión y almacenamiento de la información. En el caso de nuestra aplicación, como se puede observar en el diagrama de la Figura 1, tenemos dos servidores diferentes de los que debemos obtener y actualizar información. Por un lado, tenemos el SQL Server (ARAMIS) ya implementado en la empresa del que tan solo podemos obtener información ya que en sí este servidor contiene toda la información sobre pedidos y artículos de la empresa. Por otro lado, tenemos el MySQL Server (PRODUCCION) implementado para el correcto funcionamiento de la aplicación que se encarga de obtener y actualizar toda la información sobre los artículos en producción.

Por un lado, en la base de datos *ARAMIS* tenemos almacenada toda la información sobre los pedidos.

En primer lugar, tenemos la información sobre los artículos necesarios. Con esta información lo que haremos es ir obteniendo para cada sección cada uno de los atributos del artículo que nos interese. Por ejemplo, en la sección de Acolchado obtendremos todas las

medidas de los artículos pendientes por fabricar, posteriormente una vez elegida las medidas obtenemos la familia, y así hasta tener catalogado el producto que se desea fabricar.

En segundo lugar, tenemos la información sobre los clientes, ya que, una vez obtenido el artículo terminado el encargado debe de realizar los envíos, y para ello necesita tener información sobre para quien es cada uno de los artículos.

Por otro lado, en la base de datos *PRODUCCIÓN* tenemos toda la información necesaria para realizar la comunicación entre las diferentes secciones, además de la digitalización del Stock. Para ello tenemos una tabla de artículos pendientes para cada una de las secciones de producción y una tabla albarán donde se encuentran todos los artículos terminados. El encargado de almacén obtendrá la información del stock y de los artículos terminados para poder realizar los envíos de manera correcta.

Para obtener la información, la aplicación utiliza dos conectores correspondientes para cada uno de los servidores y tras realizar las comprobaciones de autenticación correspondientes se abre una conexión entre el servidor y la capa de aplicación para que posteriormente se realicen consultas para obtener los datos del servidor.

Con tal de tener una visión más ilustrativa, adjunto un diagrama para así comprender más fácilmente lo explicado anteriormente.

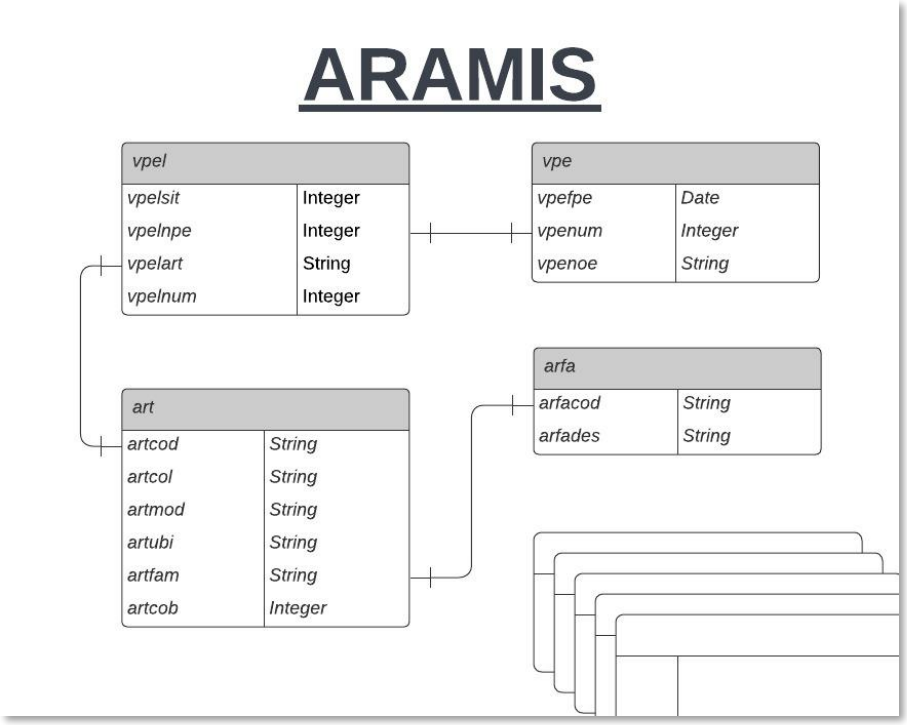

Figura 18: Diagrama ER de la base de datos de la empresa

En la Figura 18, podemos observar las tablas y entidades de la base de datos ARAMIS, se muestran únicamente las tablas y los campos que utilizamos con tal de llevar a cabo el correcto funcionamiento de la aplicación, se muestra en la esquina inferior derecha una representación de las demás tablas que existen en la base de datos ARAMIS, que no ha sido creada por nosotros sino por la empresa.

Podemos observar las siguientes entidades: Por un lado, tenemos la tabla *"vpe"* que contiene información sobre los pedidos de los clientes y su fecha de realización, entre otras muchas cosas a las que nosotros no accederemos. Esta tabla tiene una relación de 1 a 1 con el campo "*vpenum*" que está relacionado con el campo "*vpelnpe*" de la tabla "*vpel*". La tabla "*vpel*" se trata de una tabla de pedidos y artículos, en la que se organizan los artículos que se le deben servir a un pedido en concreto indicando si éste ha sido ya enviado o no. Encontramos una relación de 1 a 1 entre el campo "*vpelart*" ,que hace referencia a un artículo en concreto, con el campo "*artcod*" de la tabla "*art*". La tabla "*art*" contiene toda la información sobre los productos que se realizan en la empresa, con todas sus características, cabe mencionar la clave primaria de esta tabla "*artcod*" ya que tendrá un desempeño vital en la aplicación con tal de realizar todo el manejo de los artículos. Así pues, en esta tabla encontramos nuevamente una relación de 1 a 1 del campo "*artfam*" con el campo "*arfacod*" de la tabla "*arfa*". En la tabla "*arfa*" se almacena toda la información relacionada con las líneas de producción de manera que se relaciona una familia de producto con la línea de producción a la que pertenece.

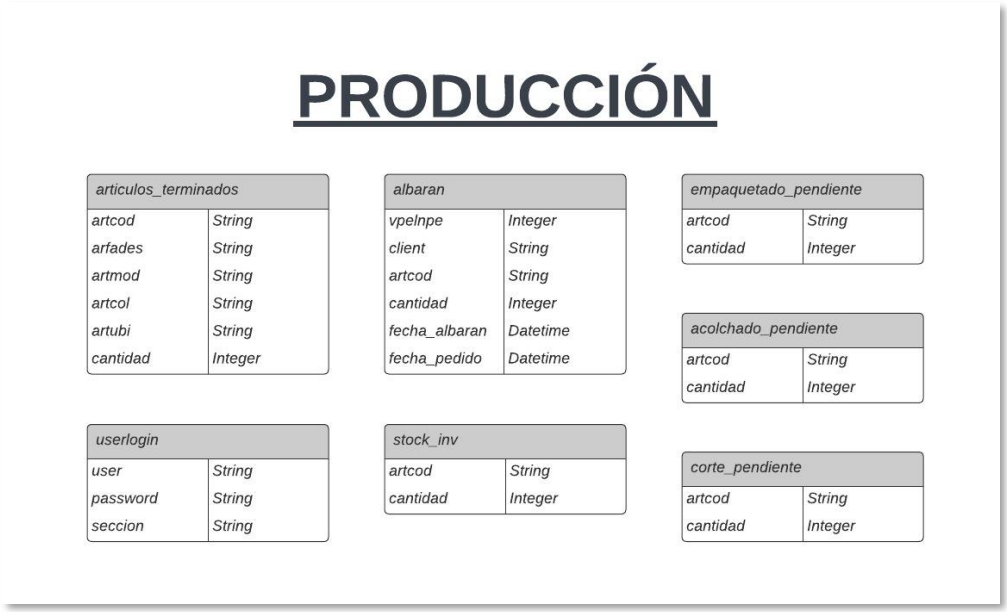

Figura 19: Diagrama ER de la base de datos de la empresa

Finalmente, en la Figura 19 se muestran las 7 tablas con las que trabajamos para mantener un control de la producción.

Primeramente, se puede observar la tabla "*userlogin*" que se encarga de realizar la autentificación y acceso del empleado a la sección que le corresponde. También tenemos todas las tablas que pertenecen a las secciones ("*corte_pendiente*", "*acolchado_pendiente*" y "*empaquetado_pendiente*") donde se almacenan todos los productos que tienen pendientes por producir, almacenando el código del producto y la cantidad. Asimismo, con tal de almacenar los productos terminados, en caso de estar solicitados por un cliente a la hora de producirlos se almacenarán en "*artículos_terminados*" y en caso de ser sobrantes se almacenarán en "*stock_inv*". Finalmente, con tal de mantener un control de los envíos y poder realizar el albarán correctamente, se mantendrá toda la información de estos en la tabla *"albaran*".

4. Tour por la aplicación

Finalmente, como no podía ser de otra forma, se realizará una demostración mediante capturas de pantalla sobre cuál es el funcionamiento de la aplicación durante su ejecución. De la misma manera que se ha procedido con anterioridad se realizarán las demostraciones de cada una de las secciones por separado.

Inicialmente, se muestra la pantalla de acceso del trabajador que es común entre todas las secciones. Mediante la que tras ingresar usuario y contraseña correctamente le dirigirá a la página inicial de la sección que le corresponda. (Fig.20)

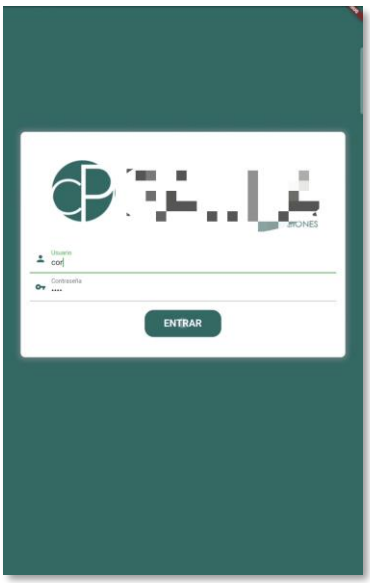

Figura 20: Captura de la aplicación que muestra el login

Con tal de que el usuario pueda ingresar su usuario y contraseña, la aplicación despliega el habitual teclado Android como se puede observar en la Figura 21.

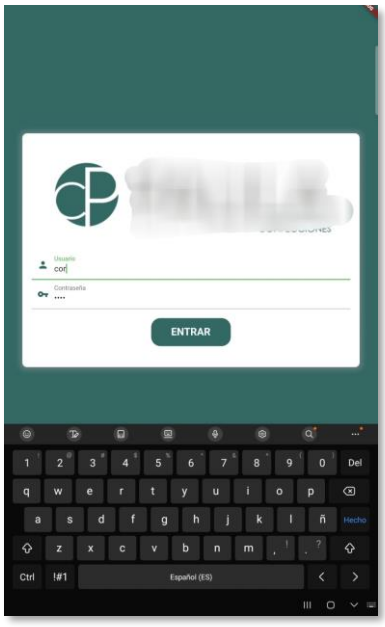

Figura 21: Captura de la aplicación que muestra el teclado que muestra el login

4.1 Sección de Acolchado

Figura 22: Captura de la aplicación que muestra la selección de medidas en la sección de Acolchado

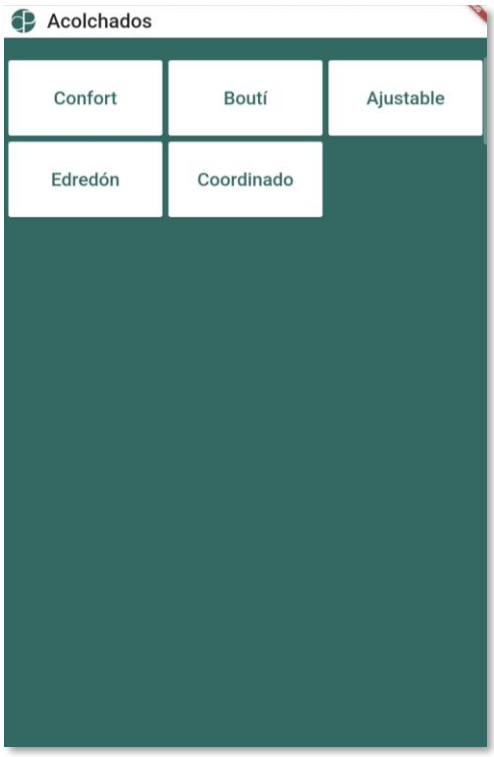

Figura 23: Captura de la aplicación que muestra la selección de familia en la sección de Acolchado

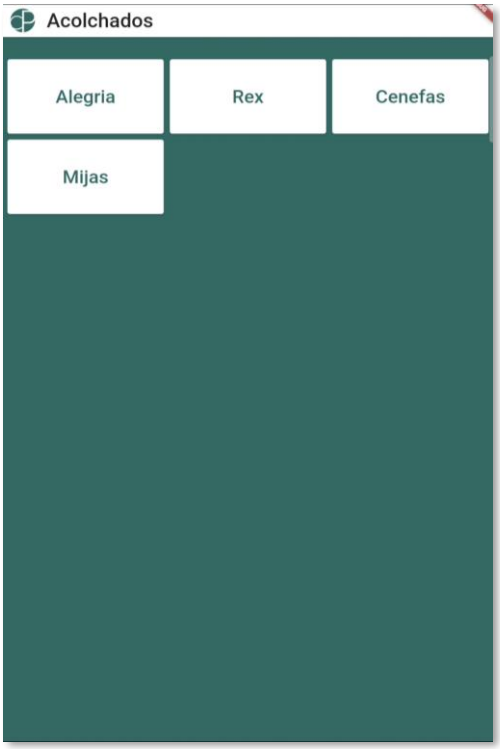

Figura 24: Captura de la aplicación que muestra la selección de modelo en la sección de Acolchado

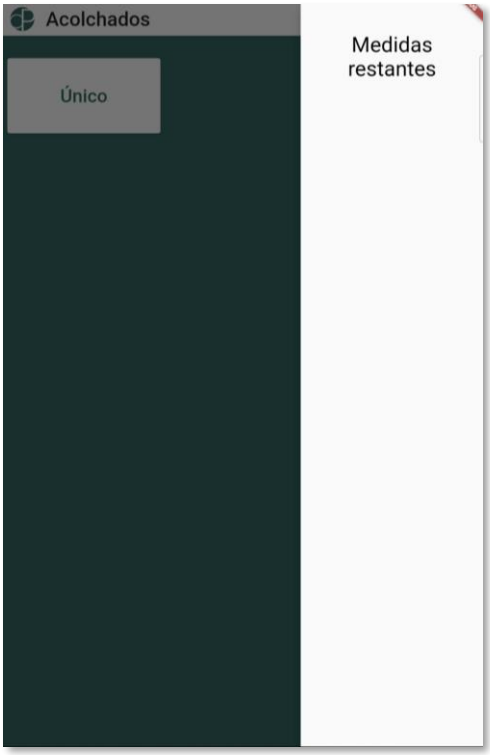

Figura 25: Captura de la aplicación que muestra la selección de color y el posterior desplegable informativo en la sección de Acolchado

Figura 26: Captura de la aplicación que muestra la confirmación del producto en la sección de Acolchado

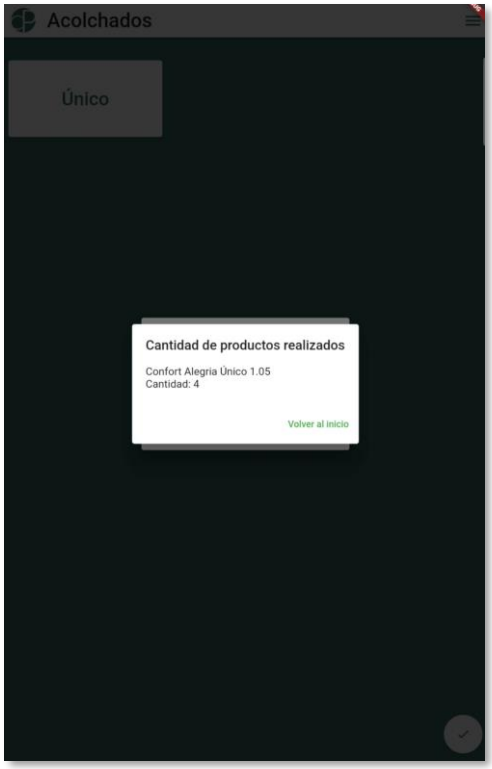

Figura 27: Captura de la aplicación que muestra la cantidad de productos que debe realizar el empleado de la sección de Acolchado

El trabajador de acolchados, una vez autenticado accede a la pantalla de selección de medidas, en ella encuentra las medidas de los pedidos hasta la fecha ordenados de manera que el más antiguo está en la parte superior izquierda. Puede seleccionar una fecha en el botón verde claro “Seleccionar fecha”, de manera que recargará las medidas hasta la fecha en la que indique, una vez pulse sobre una medida se dirigirá a la siguiente pantalla. Una vez en esta seleccionará la familia que desee, y de la misma manera, seleccionará una familia y se dirigirá a la pantalla de selección de modelo. Una vez seleccionado el modelo, se dirigirá a la última pantalla, la selección de color, en esta elegirá el color para una vez elegido el color poder realizar la confirmación del producto y proceder a su producción. Aunque, no debe proceder sin antes informarse adecuadamente sobre la existencia de otras medidas pendientes de producir del mismo modelo, que las puede consultar en el desplegable que puede consultar presionando el icono que se le muestra en la esquina superior derecha. Una vez confirmado y producido el producto se le redirige a la pantalla inicial.

4.2 Sección de Corte

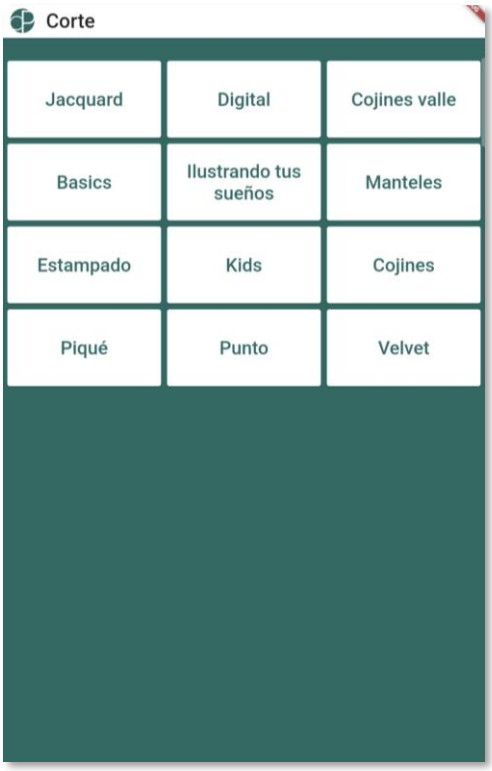

Figura 28: Captura de la aplicación que muestra la selección de línea en la sección de Corte

Figura 29: Captura de la aplicación que muestra la selección de familia en la sección de Corte

Figura 30: Captura de la aplicación que muestra la selección de modelo y color en la sección de Corte

Figura 31: Captura de la aplicación que muestra la confirmación del producto en la sección de Corte, con el aviso de “Revisa el desplegable”

Figura 32: Captura de la aplicación que muestra la confirmación del producto en la sección de Corte desde el desplegable

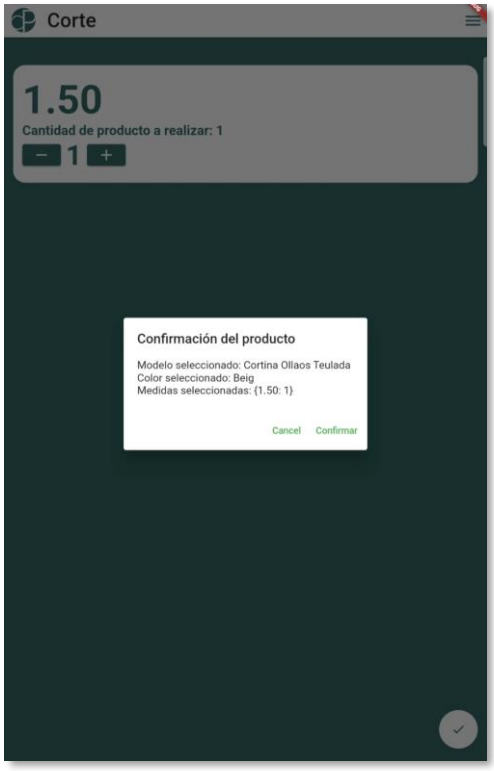

Figura 33: Captura de la aplicación que muestra como se ha redirigido al trabajador al producto con el mismo modelo y color para que así pueda trabajar de una manera más optima

El trabajador de corte inicialmente debe seleccionar una línea de producción de todas la que tenga pendientes, una vez seleccionada navega a la selección de familia donde en ella encuentra las familias de los pedidos hasta la fecha ordenados de manera que el más antiguo está en la parte superior izquierda. Puede seleccionar una fecha en el botón verde claro “Seleccionar fecha”, de manera que recargará las medidas hasta la fecha en la que indique, una seleccione una familia se dirigirá a la siguiente pantalla, la selección de modelo y color. En esta pantalla se le muestra en la parte superior un deslizable en el que se le muestran todos los modelos a seleccionar de manera que cuando pulse sobre un modelo en la parte inferior se le mostrarán todos los colores disponibles para ese modelo. Una vez seleccionado el color se dirigirá a la pantalla final. En ella encontramos una lista de medidas, con un texto que informa al trabajador de cuantos productos se deben cortar y un selector de cuantos va a producir él finalmente. Así pues, cuando se dirija a confirmar y producir el producto, la aplicación le informará de si debe consultar el desplegable en el que se le muestran otros productos para producir del mismo modelo y color, de manera que una vez produzca el artículo que estaba produciendo la aplicación le redirigirá automáticamente hasta la pantalla final de ese otro producto del mismo modelo y color, lo que le hará evitar cambiar el rollo de tela y le permitirá realizar el trabajo más rápido y eficientemente.

4.3 Sección de Empaquetado

Figura 34: Captura de la aplicación que muestra la disposición de la sección de Empaquetado

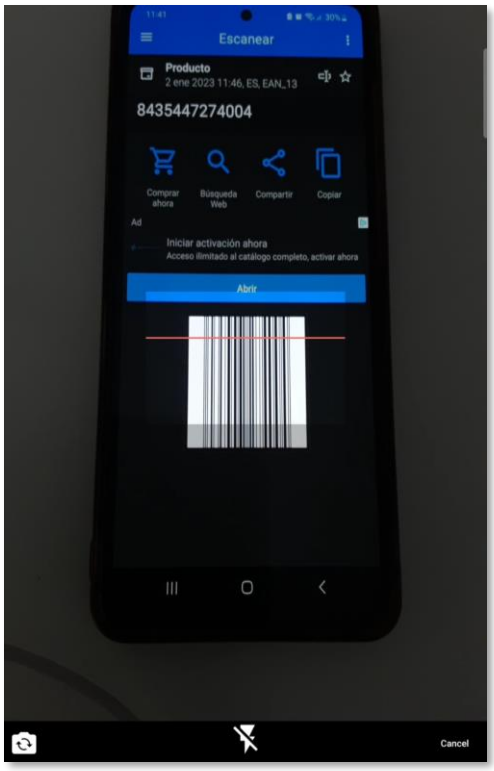

Figura 35: Captura de la aplicación que muestra el escáner de la sección de Empaquetado

Figura 36: Captura de la aplicación que muestra la lista resultante del escaneo en la sección de Empaquetado

Figura 37: Captura de la aplicación que muestra la confirmación del producto en la sección de Empaquetado

El empleado de la sección empaquetado trabaja de la siguiente manera, no navega por ninguna pantalla, simplemente, cuando le llega un producto terminado, lo deberá plegar, empaquetar y pegarle la etiqueta con el código correspondiente, es ahí donde entra en escena la aplicación, ya que el empleado deberá escanear este código con el escáner que se muestra en la Figura 35, estos código irán almacenándose en una lista que se muestra entre los dos botones disponibles en la sección. Esta lista que muestra los códigos es modificable ya que si el empleado pulsa sobre alguno de los códigos este se eliminará de la lista, para así poder corregir errores de escaneo. Finalmente, cuando el empleado decida confirmará todos los productos escaneados para que el encargado de almacén se haga cargo de ellos.

4.4 Sección de Almacén

Figura 38: Captura de la aplicación que muestra la disposición de la sección de Almacén

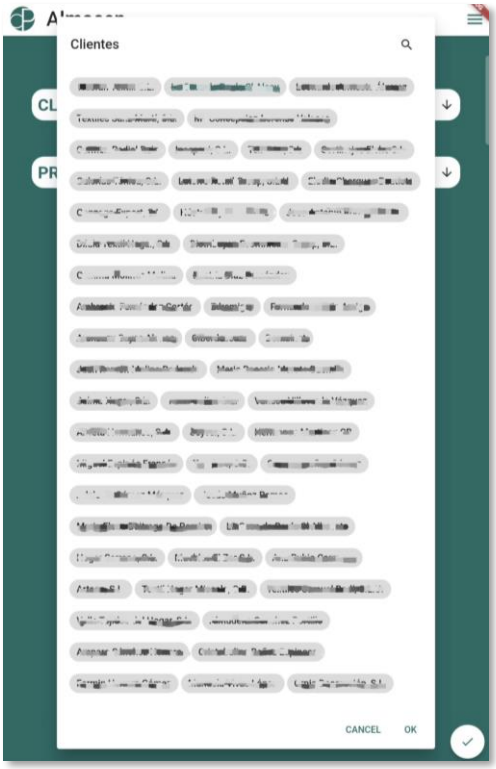

Figura 39: Captura de la aplicación que muestra la selección de clientes en la sección de Almacén

Figura 40: Captura de la aplicación que muestra la selección de productos en la sección de Almacén

Figura 41: Captura de la aplicación que muestra la selección de cliente y producto listos para el albarán en la sección de Almacén

Figura 42: Captura de la aplicación que muestra la confirmación del albarán en la sección de Almacén

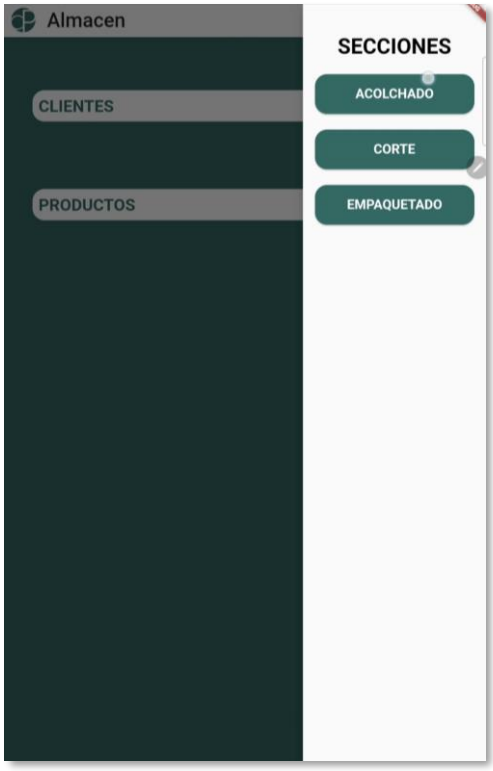

Figura 43: Captura de la aplicación que muestra el desplegable de control de las secciones en la sección de Almacén

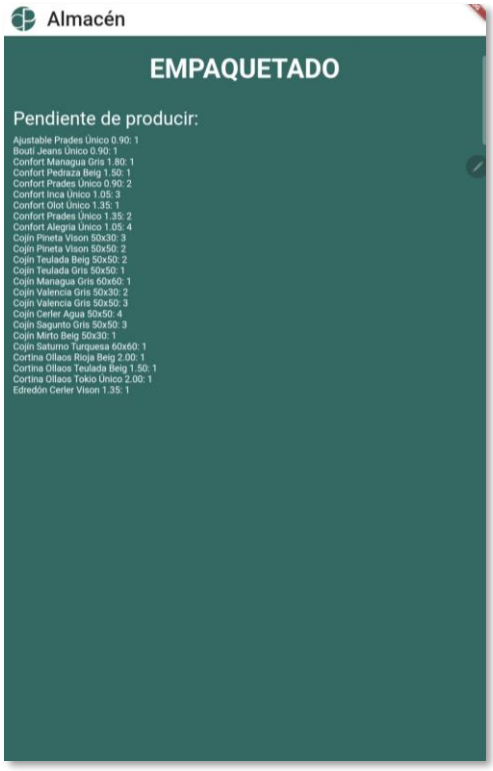

Figura 44: Captura de la aplicación que muestra los artículos pendientes de producir de la sección de Empaquetado

El encargado de almacén, con tal de realizar un envío puede proceder de dos maneras, escogiendo un cliente de la selección de clientes, de manera que la aplicación le mostrará los artículos de los que dispone para servir a ese cliente, o bien escogiendo un producto o un conjunto de productos, de manera que la aplicación la mostrará los clientes que han pedido estos artículos. Por otro lado, el encargado del almacén también tiene la capacidad de poderse informar sobre dónde se encuentran los artículos pendientes de producir, de manera que, al pulsar en la esquina superior derecha se le desplegará un *Drawer* con todas las secciones para que así pueda pulsar sobre la sección sobre la que necesita información.

Finalmente, como prueba del correcto funcionamiento de la aplicación se muestra a continuación una captura de la base de datos de Producción mostrando la tabla de "*albarán*" en la que se muestra una selección de productos que han sido albaranados y están listos para enviar.

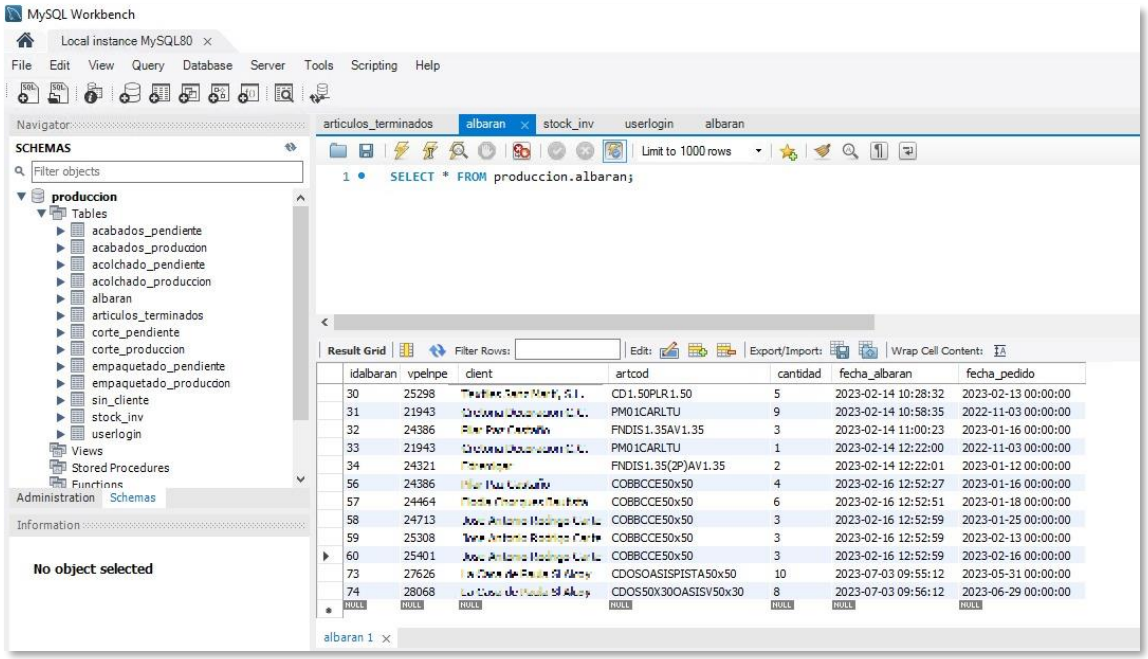

Figura 45: Captura de la base de datos de Producción donde se muestran diversos albaranes

5. Conclusiones y trabajos futuros

En resumen, el objetivo de este trabajo era poder desarrollar en su totalidad una aplicación móvil capaz de monitorizar la producción de una empresa textil, con tal de lograrlo se debía implementar una interfaz de usuario en Android con la ayuda de Flutter, un servicio RESTful en Flask en el que se desplegaría una aplicación Python y finalmente unas bases de datos capaces de trabajar con todo lo anterior. Esta aplicación debía ayudar a la producción de la empresa y provocar que los envíos mejorasen.

Tras unos duros meses de trabajo, se puede dar por cumplido el objetivo. Como resultado de ese trabajo se ha logrado obtener una aplicación totalmente funcional, capaz de controlar la línea de producción y mejorarla, una aplicación fácil de entender para los trabajadores y que están encantados de adaptar a su día a día. También cabe destacar, que no tan solo se ha conseguido una aplicación bonita, en su interior trabajan conjuntamente tanto un servidor web que maneja una aplicación en Python y dos bases de datos logrando el resultado de que, en lugar de atrasar el trabajo del empleado, logra que éste acelere.

Así pues, como tal la aplicación ha conseguido lograr que los trabajadores de la empresa se beneficien de ella, han logrado abandonar los engorros de las fotocopias que utilizaban en producción, el no saber dónde estaban los productos que necesitaban y como no, realizar los envíos de manera directa si tener que ir acumulando artículos a la espera de su envío.

Por otro lado, otro gran beneficiado de este proyecto he sido yo mismo, pues me ha ayudado enormemente a descubrir el mundo laborar y las dificultades reales que éste presenta y que uno es incapaz de encontrárselas sentado en un aula. Es por eso por lo que he podido experimentar que sucede cuando necesitamos realizar conexiones entre un móvil y un servidor web, de la misma manera como poder obtener datos de una base de datos, la importancia de la seguridad en los entornos de trabajo y un sinfín de cosas más que me han hecho mejorar muchísimo de cara a labrarme un futuro el día de mañana.

Con respecto a los trabajos futuros, hay posibles mejoras que se podrían realizar. Desde la empresa se me mostró el interés de mantener un registro con todos los movimientos que se realizaban en la aplicación y con su fecha para así controlar cuanto tardaba un trabajador en realizar qué, y poder saber si algo ocurría en cierta parte de la cadena de producción, ésta podría ser una gran mejora que añadir a la aplicación. También, como no, se podrían realizar trabajos de optimización ya que se trata de un código bastante complicado por lo que se le podría buscar soluciones más sencillas que aumentaran el rendimiento. Por otra parte, en el almacén podría ser de ayuda implementar de la misma manera que con la sección de empaquetado un lector de códigos de barras con tal de agilizar el trabajo.

No quería finalizar este trabajo sin antes agradecer a las personas que me han ayudado enormemente a lograr el objetivo, en primer lugar, al director de la empresa quien me ha hecho dar el máximo para poder lograr este proyecto y como no, a mi tutor, Javier Esparza, quien ha estado ahí siempre que he tenido cualquier problema y que no ha dudado en hacer una y mil tutorías hasta lograr sacar el proyecto adelante.

En conclusión, tras tanto trabajo y esfuerzo he logrado desarrollar una aplicación capaz de controlar la producción de una empresa, algo que a principios de carrera me hubiese parecido imposible, y así es, lo he logrado, y estoy orgulloso de ello.

6. Bibliografía

A continuación, se detallan las referencias y documentación utilizadas para las tecnologías empleadas en el proyecto:

3.11.4 Documentation. (s. f.). https://docs.python.org/3/

Aures Tic Consultors. (2020, 19 noviembre). *Aplicación para el control de la producción - Aures Tic | Odoo ERP*. Aurestic. https://aurestic.es/aplicacion-control-produccion-odoo/

cupertino_icons | Dart Package. (s. f.). Dart packages. https://pub.dev/packages/cupertino_icons

Desarrollo de APIs RESTful con Python y Flask. (s. f.). https://www.ibidemgroup.com/edu/traduccion-apis-restful-python-flask/

Drawer class - material library - Dart API. (s. f.). https://api.flutter.dev/flutter/material/Drawer-class.html

Flutter documentation. (s. f.). Flutter. https://flutter.dev/docs

flutter_barcode_scanner | Flutter Package. (s. f.). Dart packages. https://pub.dev/packages/flutter_barcode_scanner

Future class - dart:async library - Dart API. (s. f.). https://api.flutter.dev/flutter/dart-async/Future-class.html

GitLab. (s. f.). *GitLab Documentation*. https://docs.gitlab.com/

GridView class - widgets library - Dart API. (s. f.). https://api.flutter.dev/flutter/widgets/GridView-class.html

How to connect Ms SQL from a Flutter App? (s. f.). Stack Overflow. https://stackoverflow.com/questions/51294893/how-to-connect-ms-sql-from-a-flutter-app/51368593#51368593

http | Dart Package. (s. f.). Dart packages. https://pub.dev/packages/http

Markingmyname. (2023, 31 marzo). *SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS)*. Microsoft Learn. https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15

Medina, F., & Medina, F. (2022, 11 mayo). Mejores tecnologías para desarrollo de aplicaciones móviles 2022 - Armadillo Amarillo. *Armadillo Amarillo - Desarrollo mobile y web*. https://www.armadilloamarillo.com/blog/mejores-tecnologias-para-desarrollo-de-aplicaciones-moviles-2022/

Mkleehammer. (s. f.). *Home*. GitHub. https://github.com/mkleehammer/pyodbc/wiki

multi_select_flutter | Flutter Package. (s. f.). Dart packages. https://pub.dev/packages/multi_select_flutter

MySQL :: MySQL Connector/Python Developer Guide. (s. f.). https://dev.mysql.com/doc/connector-python/en/

MySQL :: MySQL Workbench Manual. (s. f.). https://dev.mysql.com/doc/workbench/en/

Parasa, S., & Parasa, S. (2021, 28 diciembre). Barcode scanner / QR Code Scanner in Flutter - FlutterAnt. *FlutterAnt*. https://www.flutterant.com/barcode-qr-code-scanner-in-flutter/

qr_code_scanner | Flutter Package. (s. f.). Dart packages. https://pub.dev/packages/qr_code_scanner

Welcome to Flask — Flask Documentation (2.0.x). (s. f.). https://flask.palletsprojects.com/en/2.0.x/