MEMORIA DEL PROYECTO

Diseño y desarrollo de una plataforma robótica para la agilización de procesos en la hostelería

Trabajo de Fin de Grado

Grado en Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

David Cruz García

Tutores

André Filipe Sales Mendes

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Certificado de los tutores

D. André Sales Mendes, D. Juan Francisco de Paz Santana y D. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

HACEN CONSTAR:

Que el trabajo titulado "DISEÑO Y DESARROLLO DE UNA PLATAFORMA ROBÓTICA PARA LA AGILIZACIÓN DE PROCESOS EN LA HOSTELERIA" ha sido realizado por D. David Cruz García, con el número de documento 70957692R y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 7 de julio de 2022

SALES MENDES ANDRE FILIPE - Y5601546W
Firmado digitalmente por SALES MENDES ANDRE FILIPE - Y5601546W
Fecha: 2022.07.07 16:58:14 +02'00'

D. André Sales Mendes

DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Firmado digitalmente por DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-76125754D, givenName=JUAN FRANCISCO, sn=DE PAZ SANTANA, cn=DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Fecha: 2022.07.07 17:23:08 +02'00'

D. Juan Francisco de Paz Santana

VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2022.07.07 17:00:59 +02'00'

D. Gabriel Villarubia González

RESUMEN

En la actualidad la falta de mano de obra en el sector hostelero, unido a problemas no tenidos antes en cuenta surgidos por la aparición de la pandemia del COVID-19, han provocado serios problemas para los establecimientos hosteleros, que han visto como las pérdidas han ido aumentando a la vez que no podían ofrecer ningún tipo de servicio por la falta de trabajadores cualificados o las restricciones sanitarias impuestas.

Por todo esto surge como necesidad para estas empresas el poder ofrecer un servicio que no dependa de un trabajador asignado para dicho trabajo y que les permita obtener ganancias con la mínima inversión posible, que requiera del mínimo mantenimiento y que sea personalizable para cada tipo de negocio donde se vaya a utilizar.

El sector hostelero es un sector muy importante en nuestro país donde se pueden ofrecer desde productos caseros hasta productos comprados en supermercados, por lo que es necesario que se ofrezca una mayor flexibilidad para el empresario a la hora de comprar el producto.

Por tanto, el objetivo principal de este proyecto de fin de grado es el diseño y la implementación de un robot que permita ofrecer un servicio de preparación de bebidas o cocteles similar al de un camarero, pero sin la necesidad de tener a una o varias personas destinadas a realizar dicho trabajo durante toda la jornada laboral. Para el control de este robot se utilizaría una aplicación web que, sin necesidad de instalación alguna, ofrecería una interfaz para la solicitud y la preparación de recetas en el caso de lo clientes como una interfaz para la gestión del robot, las bebidas y las recetas en el caso del ofertante o dueño del servicio.

Este proyecto no solo permitirá que establecimientos hosteleros con un menor poder adquisitivo puedan ofrecer un servicio a la altura. Además, no solo está dirigido hacia bares o restaurantes, si no a otro tipo de actividades como bodas o comuniones donde se podría ofrecer un servicio muy similar al de un bar en cualquier tipo de lugar.

Palabras clave: bebidas, cocteles, hostelería, preparación, robot, camarero.

SUMMARY

Currently, the lack of labor in the hospitality sector, together with problems not previously considered arising from the appearance of the COVID-19 pandemic, have caused serious problems for hospitality establishments, which have seen how the losses have been increasing at the same time that they could not offer any type of service due to the lack of qualified workers or the imposed sanitary restrictions.

For all this, it arises as a necessity for these companies to be able to offer a service that does not depend on a worker assigned for said work and that allows them to obtain profits with the minimum possible investment, that requires the minimum maintenance and that is customizable for each type of business. where it will be used.

The hospitality sector is a very important sector in our country where you can offer everything from homemade products to products bought in supermarkets, so it is necessary to offer greater flexibility to the entrepreneur when buying the product.

Therefore, the main objective of this end-of-degree project is the design and implementation of a robot that allows offering a drink or cocktail preparation service similar to that of a waiter, but without the need for one or more people. intended to perform such work during the entire working day. To control this robot, a web application would be used that, without the need for any installation, would offer an interface for the request and preparation of recipes in the case of customers as well as an interface for the management of the robot, the drinks and the recipes. in the case of the provider or owner of the service.

This project will not only allow catering establishments with less purchasing power to offer a service to match. In addition, it is not only aimed at bars or restaurants, but also at other types of activities such as weddings or communions where a service very similar to that of a bar could be offered in any type of place.

Keywords: drinks, cocktails, hospitality, preparation, robot, bartender.

Tabla de contenido

Índice de tablas

Índice de ilustraciones

1 INTRODUCCIÓN

Actualmente, en la hostelería existen diferentes maneras de preparar y consumir bebidas, licores o cocteles, y existen personas encargadas de la preparación de estas bebidas, conocidos como "*Barmans*". Sin embargo, la nueva tendencia está en usar la tecnología dentro de la hostelería como son los dispensadores automátcos de alcohol que apoyen las labores que desempeñan los "barmans".

Por ello con este proyecto se propone el diseño y desarrollo de un brazo robótico de bajo coste capaz de coger y preparar productos de un restaurante. Más concretamente se diseñará un caso de estudio centrado en la preparación de coctels y bebidas de forma autónoma por un robot. Con esto se pretende minimizar las pérdidas generadas por error en la preparación de los licores, además de permitir la preparación personalizada a gusto del consumidor.

En este documento se recogerán los aspectos más relevantes de todo el desarrollo del proyecto, desde el estudio de mercado y la planificación temporal que se ha tratado de seguir hasta los manuales de construcción y de usuario que explican el funcionamiento y resultado final del robot.

Para ello, se seguirá la siguiente estructura, recogiendo en cada punto uno o varios aspectos importantes del desarrollo del proyecto:

- ***Estudio de mercado:*** se estudiarán competidores o alternativas que existen en la actualidad en el mismo campo de desarrollo al que pertenece este sistema.
- ***Objetivos:*** se especifican los objetivos que se pretenden conseguir con el desarrollo e implementación de este proyecto, tanto personales como del sistema.
- ***Conceptos teóricos:*** se explicarán varios conceptos teóricos relacionados para facilitar la comprensión de algunos aspectos del funcionamiento y desarrollo del sistema.
- ***Herramientas y técnicas:*** herramientas y técnicas utilizadas para una correcto diseño, evolución e implementación.
- ***Aspectos relevantes:*** se explicarán algunos aspectos importantes surgidos durante el diseño e implementación de este sistema.
- ***Limitaciones de la aplicación:*** se expondrán algunas limitaciones del sistema encontradas durante y al final del desarrollo del proyecto.
- ***Líneas de trabajo futuras y conclusiones*****:** se explicarán posibles mejoras futuras encontradas y las conclusiones de todo el proceso seguido durante la realización de este proyecto.
- ***Referencias y bibliografía.***

Para completar la explicación de esta memoria, se acompañará y complementará con diferentes anexos donde se entra en detalle de forma más específica en cada una de las fases:

- ***Anexo I. Planificación temporal:*** se recogerán todos los aspectos relacionados con la estimación de tiempo y la planificación temporal de las diferentes tareas a realizar.
- ***Anexo II. Especificación de requisitos:*** enfocado a la búsqueda y elicitación de requisitos que debe cumplir el resultado final.
- ***Anexo III. Análisis de requisitos:*** se incluirá el modelo de dominio y la fase de análisis de los requisitos especificados en el documento anterior.
- ***Anexo IV. Diseño del sistema software:*** se recogerá toda la fase de diseño del sistema.
- ***Anexo V. Documentación técnica:*** permite comprender la estructura del código incluyendo la explicación y documentación de la estructura de este.
- ***Anexo VI. Manual de usuario:*** sirve como manual para los usuarios que van a usar la aplicación y el robot, explicando toda la funcionalidad del sistema para su comprensión.
- ***Anexo VII. Esquemas electrónicos, diseño y montaje del robot:*** se incluirá un manual de preparación y montaje del robot, así como los esquemas electrónicos necesarios.

2 ESTUDIO DE MERCADO

En primer lugar, antes de comenzar con la planificación del proyecto, se ha llevado a cabo un estudio del mercado disponible tratando de encontrar productos similares u orientados al mismo propósito tratando de obtener información para dirigir el proyecto hacia el desarrollo de un sistema sencillo, intuitivo y que requiera de pocos recursos.

Durante este proceso de investigación, se encontraron varios diseños y productos orientados a este propósito:

2.1 SLIPPY

Producto desarrollado por Miso Robotics para el dispensado y sellado automático de bebidas. Se puede observar el producto en la "*Ilustración 1".*

Ilustración 1. Producto "Slippy". Robot.

Este producto no fue diseñado específicamente para la preparación de cocteles, pero no solo incluye el dispensado automático de bebidas, sino que incluye auto dispensado de vasos, la posibilidad de incluir hielo y finalmente el selle hermético del vaso. Con todo esto consigue ofrecer un servicio muy completo y en un espacio muy reducido.

Para poder controlar el robot dispone de una tablet desde donde se gestionarán los pedidos a realizar, por lo que esta máquina está totalmente orientada al vendedor de las bebidas y no a la posibilidad de ser usado directamente por clientes sin necesidad de un medio humano. Esto se puede ver en la "*Ilustración 2".*

Ilustración 2. Producto Slippy. App.

2.2 BARSYS 2.0

Producto desarrollado y enfocado para la mezcla y dispensado de cocteles mediante una aplicación asociada al robot. Se puede observar el producto en la "*Ilustración 3".*

Ilustración 3. Producto Barsys 2.0. Robot

Este diseño se acerca mucho más al esperado con el desarrollo de este trabajo de fin de grado. Se dispone de varias botellas dispuestas en dispensadores que actuarán en función de las recetas registradas en el sistema.

En diferencia al anterior este robot sí está pensado para la preparación de cocteles y está controlado por una aplicación móvil asociada para la gestión total del robot, pero, al igual que el anterior, la aplicación está enfocada al comprador del robot. La app asociada al robot está dirigida hacia un único usuario que controla el sistema (el administrador), por lo que cada usuario que quiera acceder al sistema deberá disponer de dicha aplicación descargada en sus dispositivos, además de la necesidad de un código de acceso o llave para poder conectarse al robot. Esta aplicación se puede ver en la "*Ilustración 4".*

En resumen, la aplicación no diferencia los tipos de usuarios que pueden acceder al robot, administrador o clientes, ya que solo está dirigida al primero de ellos.

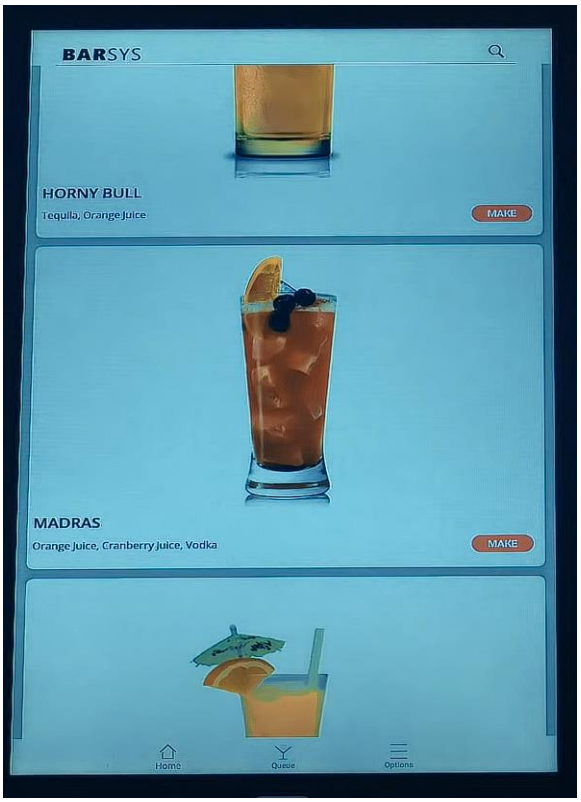


Ilustración 4. Producto Barsys 2.0. App.

3 OBJETIVOS

En este apartado se recogerán los objetivos que el sistema debe cumplir para considerarse totalmente funcional. Para ello, se definirán no solo los objetivos del sistema, sino también los objetivos personales que se pretenden alcanzar con el desarrollo de este proyecto.

3.1 OBJETIVOS PERSONALES

La motivación principal de este proyecto ha venido de la mano de una necesidad que he observado durante los meses de pandemia: la falta de camareros no humanos en el sector hostelero.

Tras observar varios proyectos y distintos sistemas que diferentes empresas están tratando de desarrollar para llevar a cabo estas tareas me entró la curiosidad de la dificultad de llevar esta tarea a cabo, pudiendo desarrollar para ello un sistema personalizado añadiendo la funcionalidad que considerase necesaria.

Además, ya que durante el grado apenas vemos contenido sobre desarrollo web (Node.js, Javascript, …), por lo que sentía una gran curiosidad por este mundo, asique decidí innovar los productos analizados durante la fase de estudio del mercado creando una plataforma web para el robot a la que pudieran acceder tanto clientes que quieren pedir una bebida como camareros/administradores que mantienen el robot y ofrecen ciertos productos a través de él.

3.2 OBJETIVOS DEL SISTEMA

Una vez determinado el tipo de proyecto a realizar, se definen una serie de objetivos de diseño, desarrollo e implementación que el sistema deberá cumplir.

El objetivo principal del robot es el de ofrecer una plataforma que prepare bebidas de forma autónoma de una manera sencilla mediante un robot de bajo coste y la máxima precisión posible.

En particular, se definen los siguientes objetivos que debe cumplir el sistema:

- ***Preparación de bebidas o cocteles:*** el sistema deberá ser capaz de preparar los cocteles según la receta indicada.
- ***Gestión del brazo robótico:*** el administrador deberá tener control total sobre el robot.
- ***Altas, bajas y modificaciones de datos de usuarios:*** se deberá permitir el alta, baja y modificación de usuarios en el sistema. En particular, estos usuarios serán clientes ya que se dispondrá de 1 único administrador en una primera instancia.
- ***Altas, bajas y modificaciones de datos de bebidas:*** el sistema permitirá el alta, baja y modificación de bebidas dentro del sistema por el administrador. Con estas bebidas se prepararán las diferentes recetas.

- ***Altas, bajas y modificaciones de datos de recetas:*** se ofrecerán diferentes recetas a los clientes que serán dadas de alta, de baja o modificadas por el administrador del sistema.
- ***Personalización en la disposición de las bebidas en el robot:*** se deberá dar la posibilidad al administrador para cambiar la disposición de las bebidas en el robot, permitiendo cambiar las botellas y ofrecer gracias a ello nuevas recetas.
- ***Almacenamiento y visualización de estadísticas:*** se deberán almacenar estadísticas sobre el número de pedidos de cada receta por cada usuario registrado. Con esto se ofrece un feedback tanto a usuarios como administrador para poder ver que tipo de cocteles se consumen más y determinar cuáles deberían estar activas.

4 CONCEPTOS TEÓRICOS

En este apartado se abordarán conceptos teóricos necesarios para el correcto entendimiento del proyecto, facilitando así la lectura y comprensión del lector.

4.1 API REST

Una API REST, o API de RESTful, es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST. Las API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la llamada) y el que requiere el productor (la respuesta).

Gracias al protocolo HTTP, se puede diseñar un API REST de manera muy sencilla, realizando el intercambio de información en XML o JSON. El modo en el que funciona una API REST puede observar en la "*Ilustración 5".*

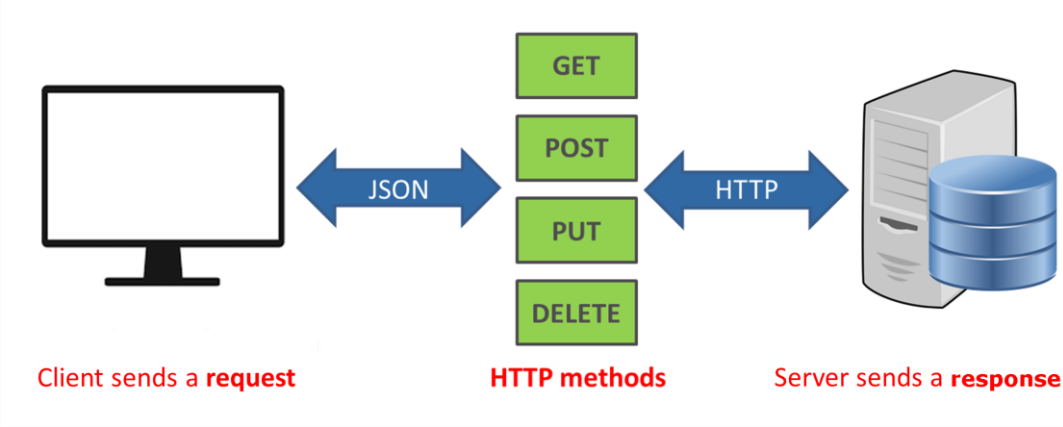


Ilustración 5. API REST

4.2 ARDUINO y RAMPS 1.4

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso. Al tratarse de código abierto, Arduino ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas, pero igualmente funcionales al partir de la misma base.

El Arduino es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que conectes se trasladará al

microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

El tipo de periféricos que puedas utilizar para enviar datos al microcontrolador depende en gran medida de qué uso le estés pensando dar. Pueden ser cámaras para obtener imágenes, teclados para introducir datos, o diferentes tipos de sensores.

También cuenta con una interfaz de salida, que es la que se encarga de llevar la información que se ha procesado en el Arduino a otros periféricos. Estos periféricos pueden ser pantallas o altavoces en los que reproducir los datos procesados, pero también pueden ser otras placas o controladores. En la "*Ilustración 6" se* pueden observar los distintos tipos de Arduino disponibles.

Ilustración 6. Tipos de Arduino.

En el caso del desarrollo de este proyecto, se ha utilizado la placa **Arduino Mega 2560,** ya que es la única placa compatible con la shield RAMPS 1.4.

El nombre RAMPS viene dado por sus siglas "*Raprap Arduino Mega Pololu Shield*". Son unos shields para Arduino MEGA diseñadas para controlar motores paso a paso, generalmente motores NEMA. Son ampliamente utilizadas para controlar impresoras 3d, ya que además de los motores paso a paso, podemos gestionar todos los periféricos que suelen componer las impresoras: ventiladores, finales de carrera, tomas de alimentación, pantalla LCD, hotends… En la "*Ilustración 7" se* puede ver la estructura de la RAMPS y sus pines y conexiones.

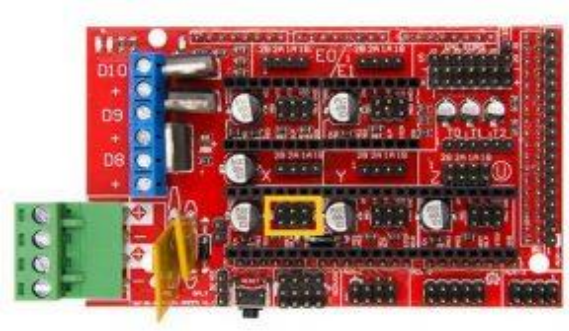


Ilustración 7. RAMPS 1.4.

4.3 MOTOR DE PASOS

Los motores de pasos son motores de corriente continua que convierten energía eléctrica en mecánica provocando un movimiento rotatorio por la acción de un campo magnético. Son ampliamente utilizados en el diseño de impresoras 3D y otros dispositivos de precisión. El funcionamiento de los pasos de un motor de pasos se muestra en la "*Ilustración 8*".

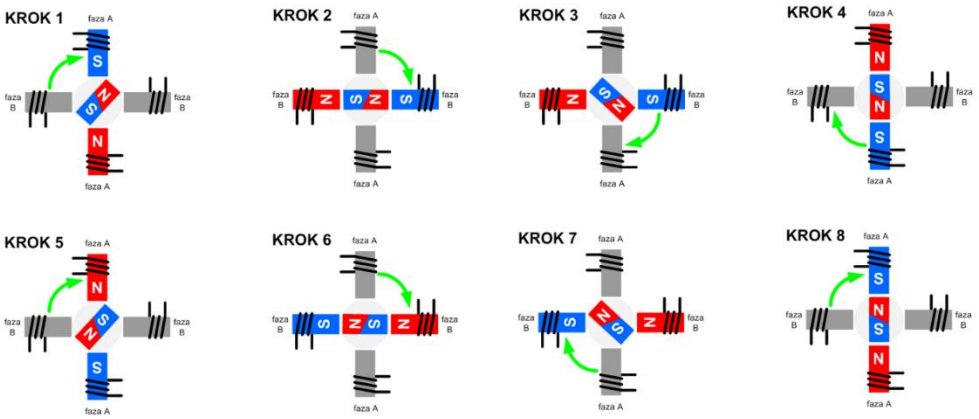


Ilustración 8. Pasos de un motor de pasos.

Un motor está compuesto de 4 partes printipalmente:

- ***Estator***: La parte fija del motor, está compuesto por un electroimán formado por un número par de polos. Las bobinas que los contienen son las encargadas de producir el campo inductor al circular por ellas la corriente de excitación.
- ***Rotor***: La parte giratoria, formada por núcleo magnético alrededor del cual va el envuelto de inducido, sobre el cual actúa el campo magnético.
- ***Colector de delgas***: Un anillo de láminas de cobre llamadas delgas, están colocadas sobre el eje del motor que sirven para conectar las bobinas del inducido con el circuito a través de las escobillas.
- ***Escobillas***: Son unas piezas de grafito que se colocan sobre el colector de delgas, permitiendo la unión eléctrica de las delgas con los bornes de conexión al inducido.

Podemos observar las distintas partes del motor descritas en la “*Ilustración 9”*.

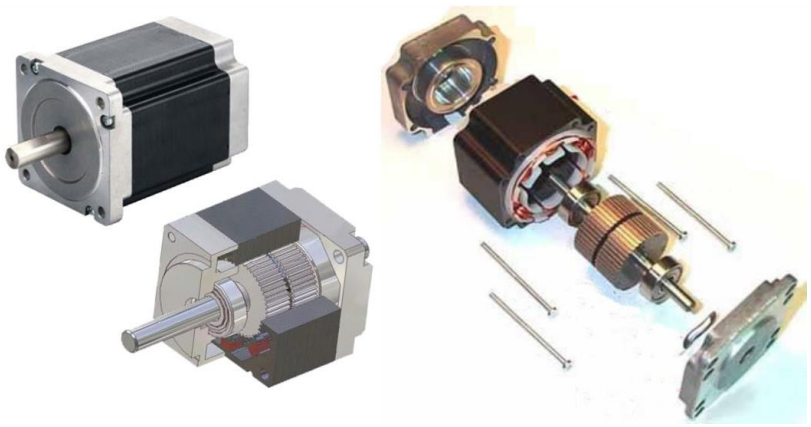


Ilustración 9. Motor de pasos.

5 HERRAMIENTAS Y TÉCNICAS EMPLEADAS

En este apartado se presentarán y explicarán las herramientas de desarrollo, bibliotecas o módulos y otras técnicas utilizadas para el desarrollo de este proyecto. Para ello, se dividirá en 2 bloques:

- Herramientas utilizadas para el desarrollo de los servidores web, tanto el servidor Frontend como el servidor Backend.
- Herramientas utilizadas para el diseño, construcción y manejo del robot.

5.1 HERRAMIENTAS EMPLEADAS EN LOS SERVIDORES

5.1.1 HTML, CSS y JAVASCRIPT

HTML (*"HyperText Markup Language"*) es un lenguaje de marcado de hipertexto y es el componente básico de una página web. Este lenguaje utiliza marcas para etiquetar texto, imágenes y otros contenidos para mostrarlos en el navegador web. Las palabras clave de su nombre son:

- **HyperText**: hace referencia a los enlaces entre las diferentes páginas que componen un proyecto web.
- **Markup**: hace referencia al uso de marcas para anotar el texto, imágenes u otros elementos.

CSS (*"Cascading Style Sheets"*) es un lenguaje que maneja el diseño y la presentación de las páginas web mediante hojas de estilo. Este lenguaje da estilo y apariencia a las estructuras creadas mediante le lenguaje HTML.

Javascript es un lenguaje de programación que permite implementar funciones complejas en el desarrollo de páginas web del lado del cliente junto a HTMl y CSS, ofreciendo un amplio abanico de opciones como mapas interactivos, gráficos 2D o 3D, etc.

Aunque es un lenguaje orientado al lado del cliente (*Dynamic client-side scripting)* existen frameworks que permiten generar scripts del lado del servidor, ofreciendo así la opción de crear un servidor web utilizando para ello Javascript.

Este lenguaje se caracteriza por:

- Lenguaje del lado del cliente
- Lenguaje orientado a objetos
- De tipado débil: no es necesario especificar el tipo de dato de ninguna variable
- De alto nivel: ofrece funciones de alto nivel con una sintaxis muy sencilla.
- Interpretado: el navegador web será quien interprete las líneas de código de un fichero Javascript.

Estos lenguajes trabajan de forma conjunta para lograr un diseño único y con funcionalidad en un entorno web. Se puede observar la forma en la que los 3 lenguajes se combinan en la "*Ilustración 10"*.

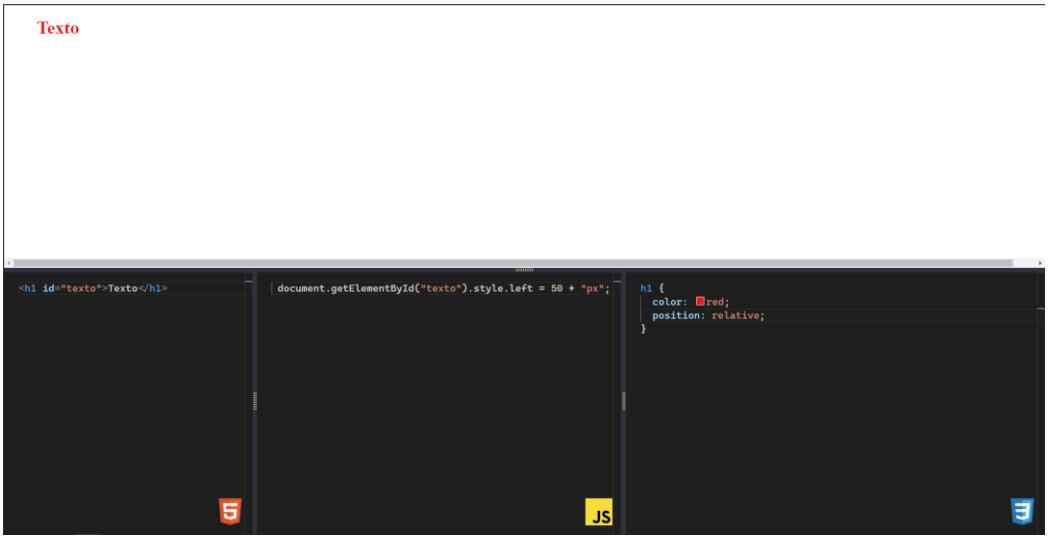


Ilustración 10. Interfaz programación HTML-CSS-JS

5.1.2 NODE JS

Node.js es un entorno multiplataforma de tiempo de ejecución que incluye todo lo necesario para ejecutar un programa escrito en Javascript sin necesidad de un navegador del lado del cliente para interpretarlo.

Este entorno de Javascript incluye una serie de módulos básicos que permiten desarrollar diferentes funcionalidades para trabajar con ficheros, crear servidores web, etc. Además, Nodejs incluye una herramienta denominada "NPM" (*Node Package Manager*). Esta herramienta es un gestor de módulos de Node.js que permite la instalación/desinstalación de módulos de terceros de una forma muy sencilla mediante un comando.

La principal ventaja de Nodejs es la gran cantidad de módulos desarrollados para este entorno, pudiendo ofrecer con ellos casi cualquier tipo de funcionalidad de una forma rápida y segura. En mi caso, se utilizaron varios módulos para trabajar con imáges, rutas, bases de datos, etc.

Como además se trata de un entorno multiplataforma, será muy sencillo trasladar un servidor creado con Nodejs de una plataforma a otra ya que será compatible en ambas.

5.1.3 EXPRESS

Express es un framework web que se aloja dentro de un entorno de ejecución Node.js. Ofrece un entorno de trabajo para construir aplicaciones web y APIs ya que proporciona funcionalidades como enrutamiento de peticiones, gestión de sesiones,

etc. Además, Express permite configurar middlewares para recibir/responder a peticiones HTTP.

En mi caso, se utilizará para el desarrollo de la API ofrecida en el servidor Backend.

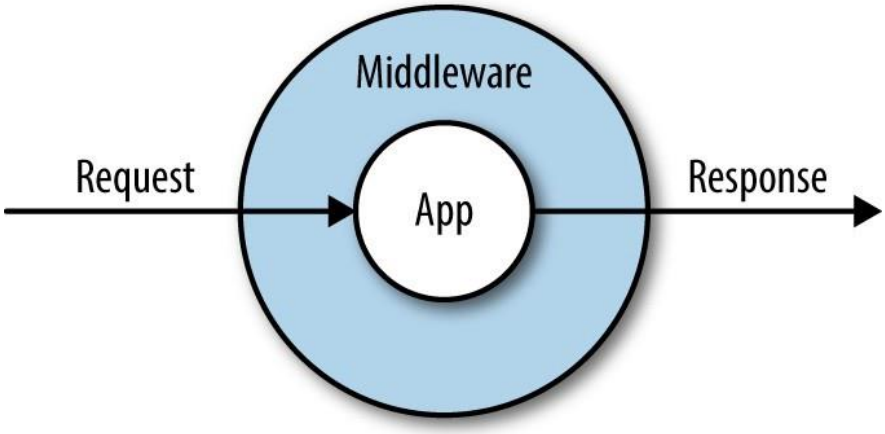


Ilustración 11. Express middleware.

5.1.4 REACT

React es una biblioteca de Javascript de código abierto mantenida por Facebook y la comunidad de software libre, y diseñada para crear interfaces de usuario mediante el uso de componentes, facilitando el desarrollo de aplicaciones web. En react el elemento principal es el componente: elementos autónomos que sirven como pequeñas piezas de código fáciles de usar y reutilizar. Esta biblioteca se puede integrar de manera sencilla con un entorno Node.js.

Sus características son:

- Composición de componentes
- Elementos y JSX: jsx es la extensión utilizada para los archivos en react, definiendo una sintaxis propia similar a HTML en Javascript.
- Componentes con y sin estado
- Permite el desarrollo de aplicaciones móviles

5.1.5 MATERIAL UI

MaterialUI es una biblioteca Javascript de código abierto que implementa es diseño "material" de Google en componentes de React. Se pueden ver algunos de estos componentes en la "*Ilustración 12".*

Ofrece la posibilidad de combinar una biblioteca gráfica con una gran cantidad de componentes con React. Con esto se consigue una gran variedad de componentes prediseñados siguiendo la filosofía "material", acelerando el desarrollo del Frontend de nuestro servidor web.

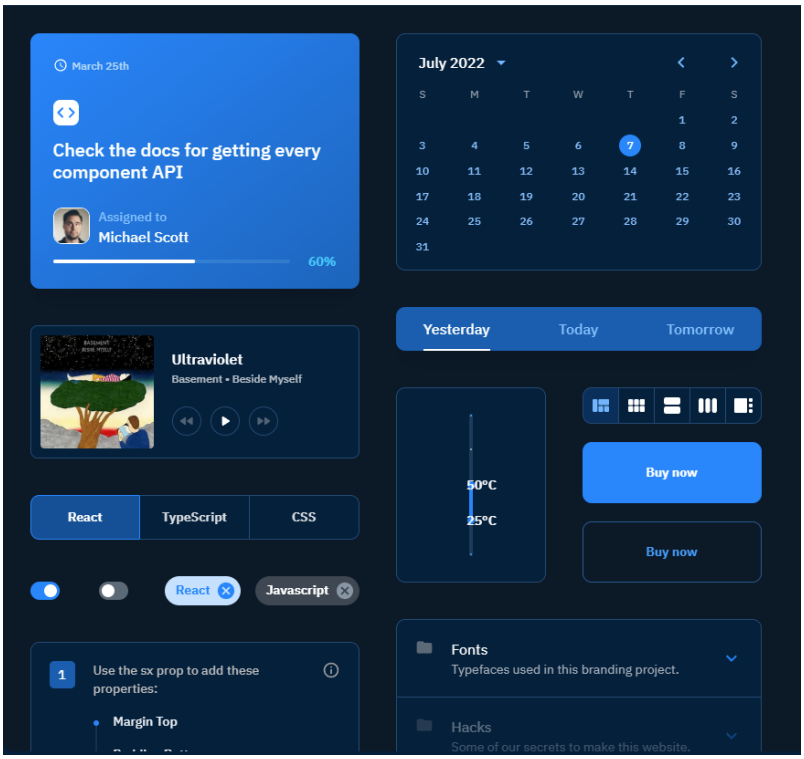


Ilustración 12. Ejemplo componentes MaterialUI

5.1.6 PLANTILLA WEB MATERIAL DASHBOARD 2

Plantilla desarrollada utilizando MaterialUI que ofrece una gran variedad de componentes que se usarán para la creación de la página web de administración del robot en el servidor Frontend. Está protegida con una licencia MIT que permite su uso, copia, modificación, publicación y venta mientras se mencione al autor de la plantilla: Creative Tim.

Esta plantilla se utilizará como base para la web de administración ya que proporciona una web de ejemplo con un menú lateral y varias rutas donde se construirá toda la página web a desarrollar. Además, la plantilla es compatible con MaterialUI, por lo que los componentes podrán ser utilizado sin ningún tipo de problema.

Esta plantilla presenta un diseño muy sencillo con una barra de navegación que utilizaré para presentar las distintas opciones al usuario (se puede ver en la "*Ilustración 13*").

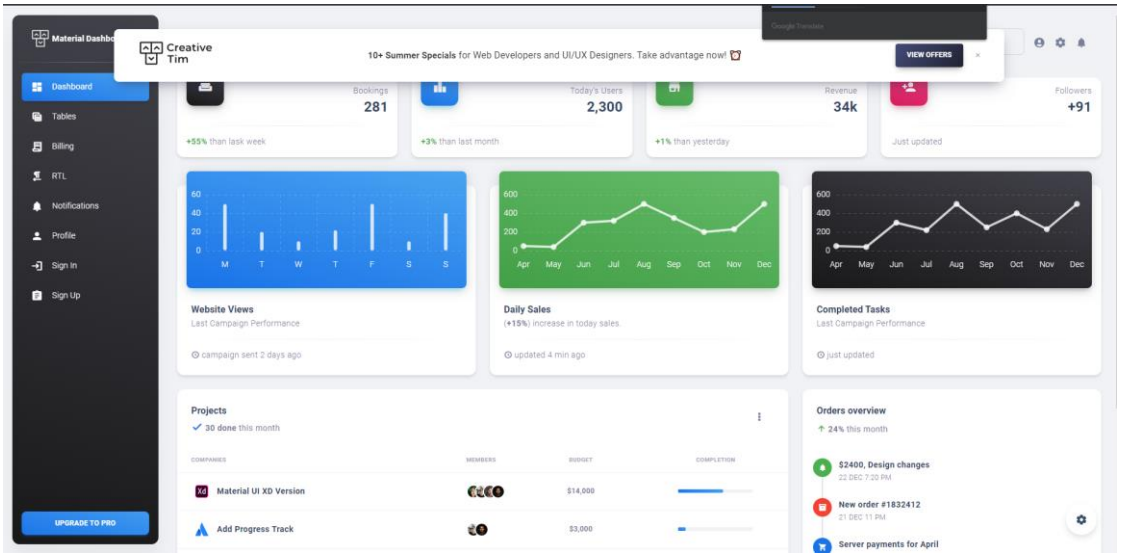


Ilustración 13. Plantilla web Material Dashboard 2

5.1.7 MYSQL/SEQUELIZE

MySQL es un sistema gestor de bases de datos relacionales desarrollado por Oracle. MySQL utiliza múltiples tablas dentro de una base de datos donde se almacenará y organizará la información necesaria para el correcto funcionamiento del sistema garantizando la persistenia de los datos.

Además, permite una fácil integración con todo tipo de lenguajes de programación: en especial, se dispone de un módulo NPM (Sequelize) para NodeJS con el que podremos acceder y realizar peticiones a esta base de datos.

Para la conexión a la base de datos se utilizará el módulo *Sequelize* de NodeJS que permite realizar consultas de una manera muy sencilla a la base de datos.

5.1.8 APACHE/PHP/PHPMYADMIN

Para la correcta gestión de la base de datos, se ha utilizado un servidor web Apache y la herramienta PhpMyAdmin para poder conectarme a la base de datos y consultar las tablas creadas y los datos almacenados. Parar poder trabajar con PhpMyAdmin es necesario tener PHP instalado en el dispositivo.

PhpMyAdmin presenta un panel de control visual con el que se podrá trabajar de forma cómoda, pudiendo incluso ver el diseño relacional de la base de datos MySQL (esto se puede ver en la "*Ilustración 14").*

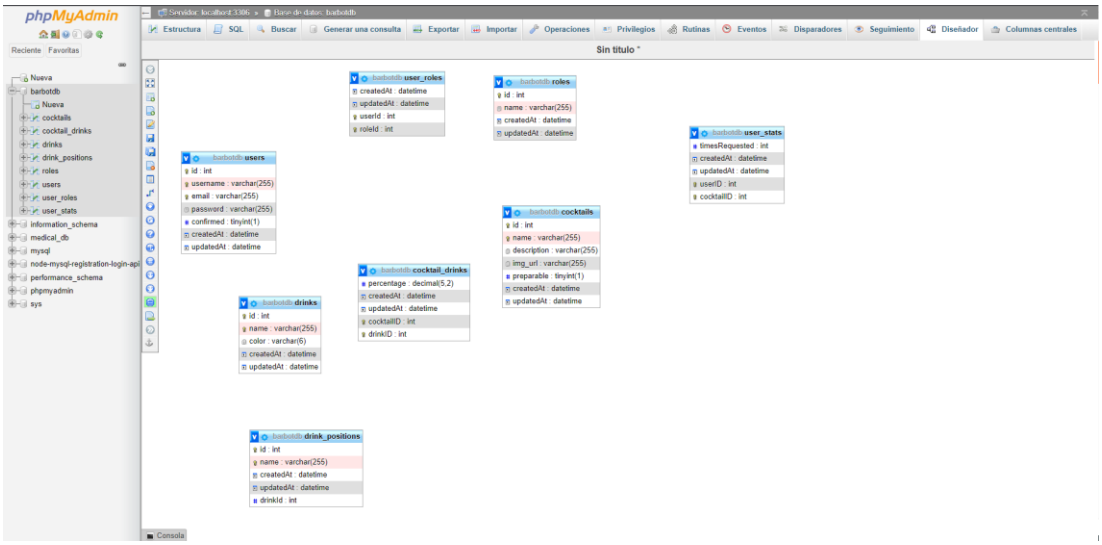


Ilustración 14. Interfaz PHPMyAdmin.

Esta herramienta solo se utiliza durante el desarrollo del sistema, no es necesario disponer de ella para el uso de la aplicación web o del robot.

5.1.9 POSTMAN

Herramienta orientada a la realización de peticiones HTTP a una API para testing. Con esta herramienta se puede comprobar si cada endpoint de la API REST desarrollada trabaja correctamente y sin ningún error.

PostMan ofrece una interfaz muy sencilla, que se puede observar en la "*Ilustración 15",* con la que el usuario trabajará introduciendo la URL y generando la petición.

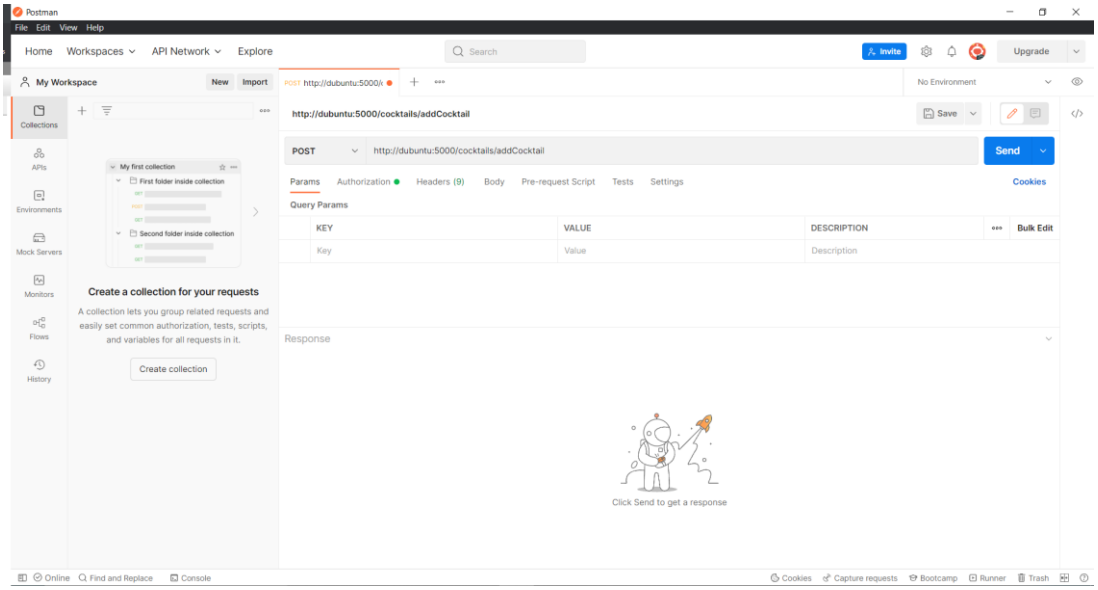


Ilustración 15. Interfaz PostMan.

5.1.10 VISUAL STUDIO CODE

Editor de código que incluye una gran cantidad de funcionalidad y extensiones que permiten integrar un proyecto con Git de forma visual, permiten acceder y mostrar bases de datos o incluso ayudas de autocompletado. Soporta gran cantidad de lenguajes con muchas extensiones para ellos, por lo que es el entorno perfecto para desarrollar la aplicación. Su diseño se puede ver en la "*Ilustración 16".*

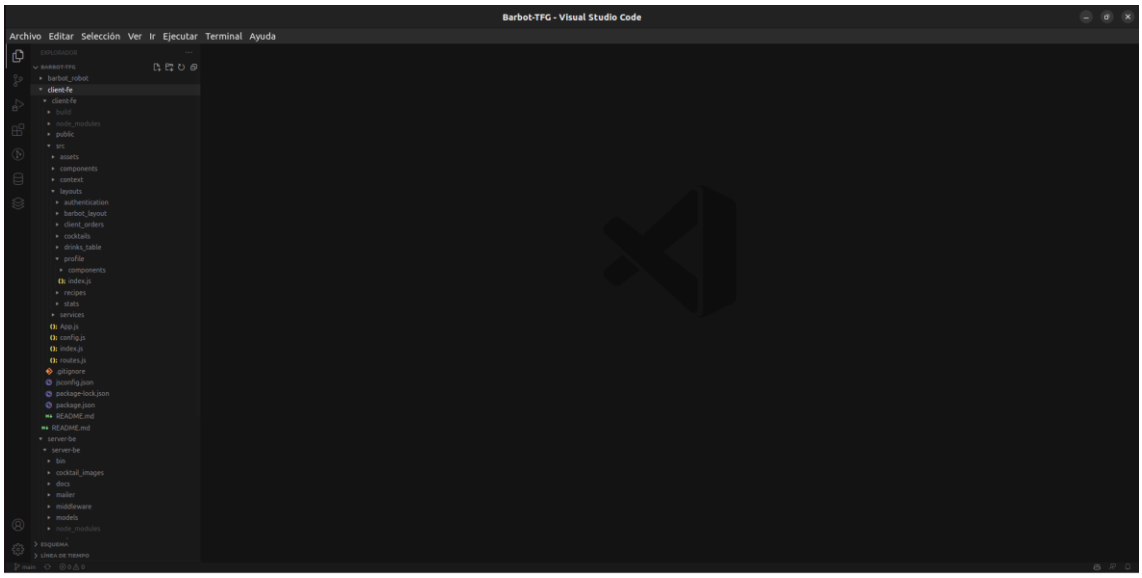


Ilustración 16. Interfaz Visual Studio Code.

5.2 HERRAMIENTAS EMPLEADAS EN EL ROBOT

5.2.1 MARLIN

Firmware libre diseñado para el control de impresoras 3D mediante el uso de comandos denominados GCODE recibidos por el puerto serie de la placa Arduino donde se debe instalar. Dispone de una gran cantidad de opciones definidas para configurar las impresoras 3D.

En mi caso, se han modificado algunas opciones de este firmware para el correcto funcionamiento de los motores de pasos que controlan el movimiento del robot, así como el control de los finales de carrera o el LED de aviso del robot (esta versión modificada del firmware se adjuntará junto al código).

5.2.2 ARDUINO / ARDUINO IDE

Arduino IDE es un editor de código fuente específico para el desarrollo de código escrito en lenguaje Arduino. El lenguaje Arduino no es más que una extensión de C++ incluyendo librerías que facilitan el desarrollo de programas que se cargarán en una placa Arduino.

Un programa Arduino tiene una estructura muy sencilla dividida en 2 bloques (se muestran en la "*Ilustración 17"*:

- ***Setup:*** define la preparación del programa (declaración de variables, inicializaciones, …). Es la primera función en ejecutarse y se ejecuta 1 sola vez.
- ***Loop:*** bucle de ejecución del programa que se repite continuamente. Aquí se realizarán las acciones necesarias para esperar las órdenes enviadas desde la web y transformadas en GCODE por Octoprint.

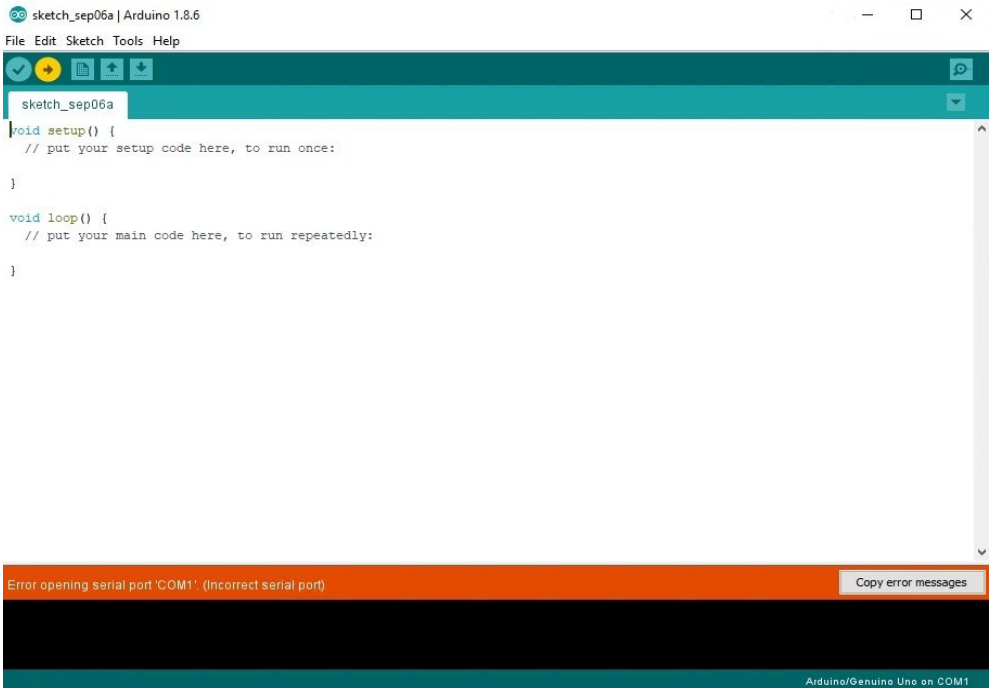


Ilustración 17. Interfaz Arduino IDE

5.2.3 OCTOPI / OCTOPRINT API

Aplicación para controlar impresoras 3D. Dispone de un sistema operativo personalizado, instalable en un dispositivo ARM como una RaspberryPI, ofreciendo una página web y una API (Octoprint API) para la comunicación con la impresora 3D.

En mi caso, aunque dispone de muchas opciones para trabajar con impresoras 3D, solo se utilizará la web para ver el estado del robot (se muestra en la "*Ilustración 18"*) y la API para enviar los comandos necesarios para el movimiento del robot.

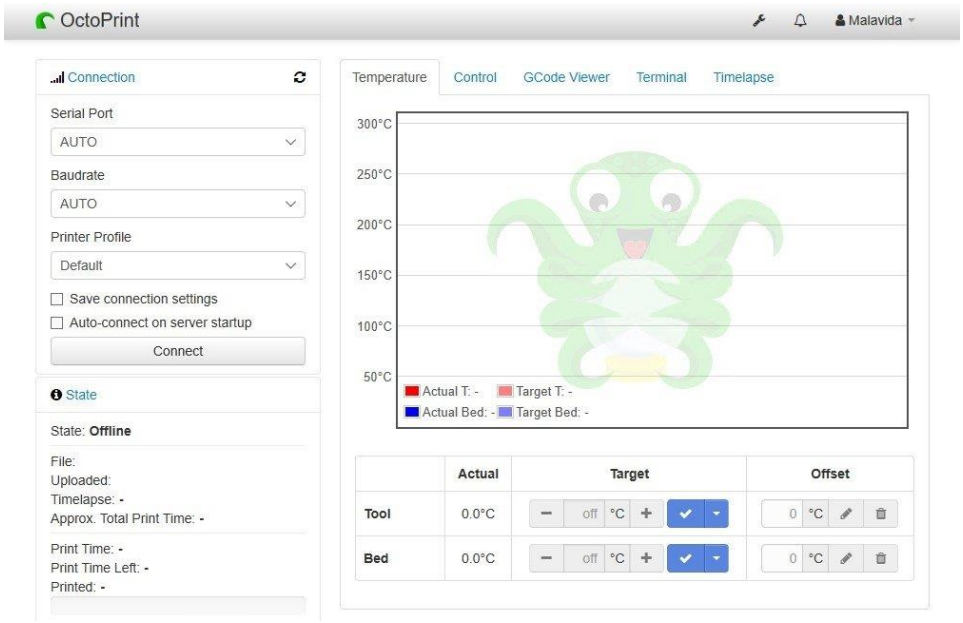


Ilustración 18. Interfaz OctoPrint.

5.2.4 PRONTERFACE

Es un programa de impresión 3D utilizado para el control manual de una impresora 3D. En mi caso fue usado en la fase de pruebas del robot para determinar si los motores y las conexiones realizadas funcionaban de manera correcta o no. Presenta una interfaz con la que se pueden controlar de forma manual cada motor (“*Ilustración 19”)*.

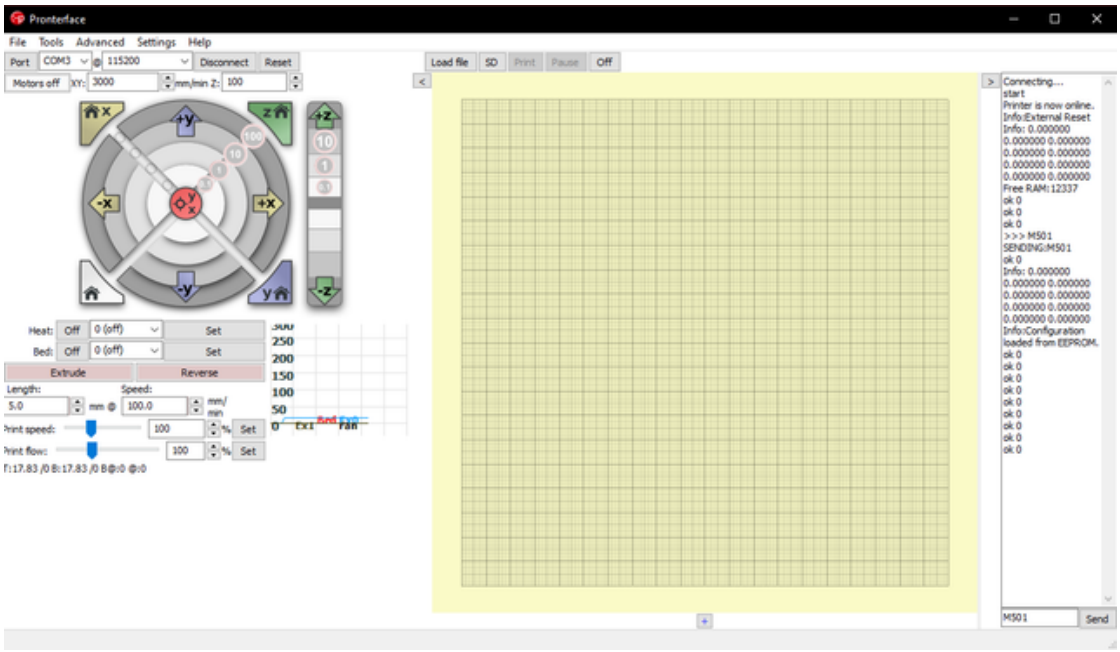


Ilustración 19. Interfaz Pronterface.

5.2.5 SKETCHUP

Es un programa de diseño gráfico y modelado 3D. Se ha utilizado para el diseño del robot para su posterior construcción y la modificación de algunas de las piezas 3D empleadas para la construcción del robot. Su interfaz puede ser observada en la "*Ilustración 20*".

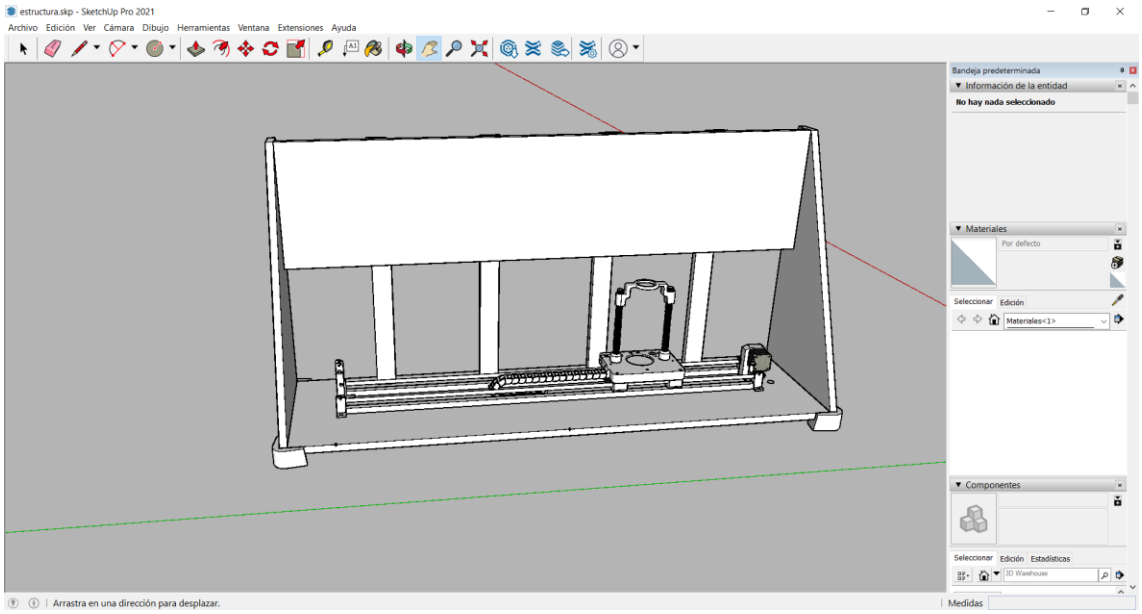


Ilustración 20. Interfaz Sketchup.

5.3 HERRAMIENTAS DE CONTROL DE VERSIONES

5.3.1 GIT

Software de control de versiones, diseñado principalmente para el desarrollo de Linux, por lo que trabaja en la eficiencia, compatibilidad y confiabilidad.

Git trabaja con repositorios de manera distribuida, por lo que cada desarrollador implicado mantendrá una copia, actualizada o no, del repositorio hasta actualizarlo respecto a los demás.

Sus características son:

- Muy potente y es de software libre
- Distribuido, no hay un repositorio central
- Permite el trabajo en diferentes ramas (ver *"Ilustración 21"*), manteniendo correctamente un historial completo de todas las versiones almacenadas.

5.3.2 GITHUB

Plataforma que sirve para alojar repositorios del sistema de control de versiones Git. En mi caso se ha utilizado para mantener el repositorio de forma privada en un

host remoto, permitiéndome trabajar desde varias localizaciones y siempre disponiendo de los últimos cambios realizados.

Ilustración 21. Trabajo con el uso de ramas de Git en Github.

5.4 HERRAMIENAS DE DOCUMENTACIÓN

5.4.1 JSDOC

JSDoc es un generador de documentación para Javascript. Analiza el código buscando comentarios personalizados para ello y genera una página HTML según estos comentarios, creando una documentación con ella. Esta herramienta se ha utilizado para generar la documentación del código implementado para los servidores.

5.5 HERRAMIENTAS CASE

5.5.1 DRAW.IO

Software de dibujo de gráficos de código abierto diseñado para diagramas de todo tipo. En especial, se ha utilizado para crear los diagramas de clases, de secuencia, … incluidos en los anexos en las fases de análisis de requisitos y de diseño (ver *"Ilustración 22"*).

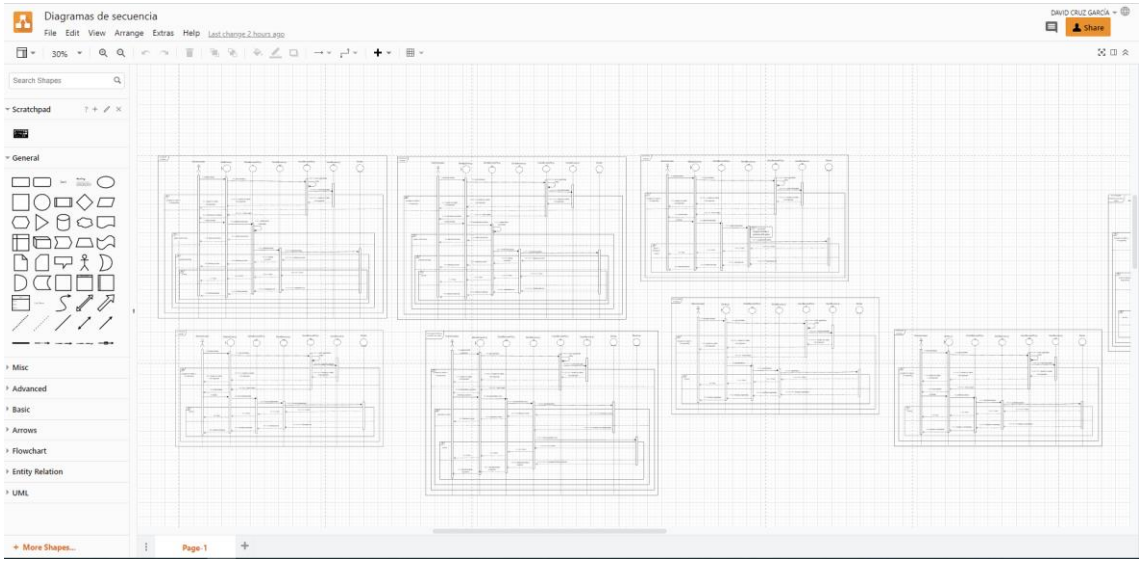


Ilustración 22. Interfaz Draw.io

5.5.2 EZ ESTIMATE

Herramienta utilizada para estimar la duración de un proyecto en función de los puntos de caso de uso, su complejidad, los actores que interaccionan con el sistema y otros factores (de complejidad y de ambiente). Puede observarse su interfaz en la "*Ilustración 23")*

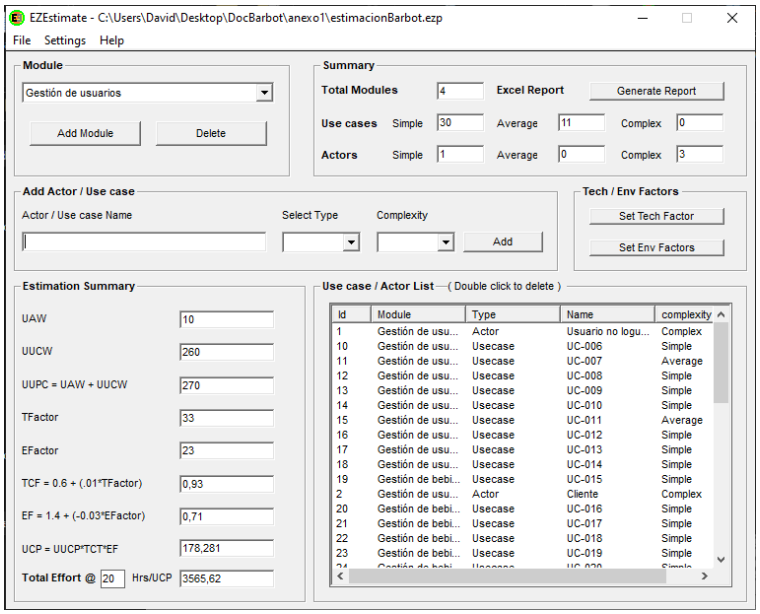


Ilustración 23. Interfaz EZ-Estimate

5.5.3 MICROSOFT PROJECT

Herramienta utilizada para planificar temporalmente las tareas a realizar (ver *"Ilustración 24")*, asignando tiempos y recursos, tratando de obtener un tiempo similar al estimado. Permite la realización de planes de proyecto, su seguimiento, construcción de diagramas de Gantt, etc.

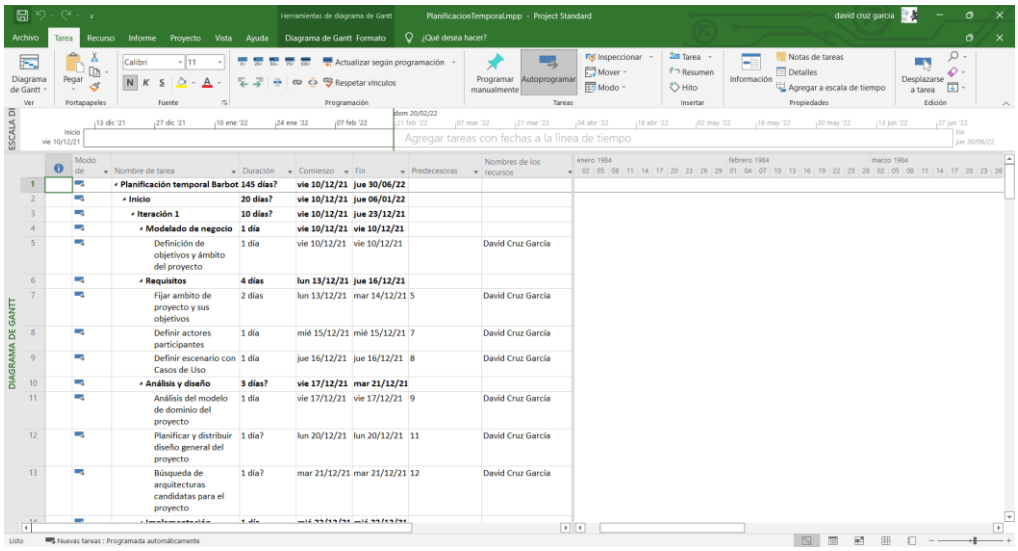


Ilustración 24.Interfaz Microsoft Project.

6 ASPECTORES RELEVANTES DEL PROYECTO

En este apartado se explicarán las partes más importantes del desarrollo del proyecto, desde la fase inicial de estimación de duración del proyecto hasta la fase final de implementación. Para ello, se definirá en primer lugar el marco de trabajo a seguir.

6.1 MARCO DE TRABAJO

El marco de trabajo utilizado ha sido el Proceso Unificado que define un marco de desarrollo guiado por lo casos de uso, siguiendo un desarrollo iterativo e incremental.

Sus principales características son:

- ***Dirigido por CU:*** los casos de uso capturan los requisitos funcionales, sus relaciones y además definen el contenido en las iteraciones definidas.
- ***Centrado en arquitectura:*** pueden existir diferentes modelos o vistas que definan la arquitectura del software del sistema, no hay un único modelo que cubra todas las necesidades.
- ***Iterativo e incremental:*** el Proceso Unificado está compuesto de 4 fases, explicadas a continuación, que están divididas en una serie de iteraciones. Cada una de estas iteraciones da como resultado un incremento en el desarrollo del producto final, añadiendo mejoras de funcionalidad o eliminando errores de iteraciones anteriores.

Como se acaba de comentar, el Proceso Unificado se divide en 4 fases que a su vez estarán divididas por distintas iteraciones. Estas 4 fases son:

- ***Inicio:*** fase inicial del proyecto donde se define el modelo de negocio del sistema, guiado por los costes, las metas, etc, según los requisitos de los clientes.
- ***Elaboración:*** En esta etapa de reajustan las estimaciones realizadas en la etapa anterior y se añaden nuevos requisitos.
- ***Construcción:*** Evolución del sistema desde sus primeros pasos hasta finalizar un prototipo que incluye los requisitos mínimos acordados.
- ***Transición:*** fase final del proyecto donde el sistema ya debe estar listo para probarse, instalarse y usarse por el usuario final.

Estas fases del desarrollo software del proyecto y sus iteraciones pueden ser representadas mediante el siguienet diagrama:

Ilustración 25. Fases e iteraciones del Proceso Unificado.

Como se puede observar en la *Ilustración 25*, se puede ver que cada una de las 4 fases está a su vez dividida en diferentes iteraciones.

Una vez definido el marco de trabajo donde se va a desarrollar el proyecto, se procede a realizar la planificación temporal del proyecto.

6.2 ESTIMACIÓN TEMPORAL DEL PROYECTO

Para comenzar con el proyecto, la primera tarea fue realizar una estimación del tiempo de duración del proyecto para, posteriormente, planificar las diferentes tareas a realizar de una manera óptima y siempre dentro de esta estimación.

Para obtener más información sobre este apartado, consultar el anexo "*Anexo I. Plan de proyecto".*

Para realizar esta estimación se utilizó la herramienta EZEstimate (se pueden observar las figuras *"Ilustración 26"* y *"Ilustración 27"*).

Use case / Actor List (Double click to delete)

Id	Module	Type	Name	complexity
1	Gestión de usu...	Actor	Usuario no logu...	Complex
10	Gestión de usu...	Usecase	UC-006	Simple
11	Gestión de usu...	Usecase	UC-007	Average
12	Gestión de usu...	Usecase	UC-008	Simple
13	Gestión de usu...	Usecase	UC-009	Simple
14	Gestión de usu...	Usecase	UC-010	Simple
15	Gestión de usu...	Usecase	UC-011	Average
16	Gestión de usu...	Usecase	UC-012	Simple
17	Gestión de usu...	Usecase	UC-013	Simple
18	Gestión de usu...	Usecase	UC-014	Simple
19	Gestión de bebi...	Usecase	UC-015	Simple
2	Gestión de usu...	Actor	Cliente	Complex
20	Gestión de bebi...	Usecase	UC-016	Simple
21	Gestión de bebi...	Usecase	UC-017	Simple
22	Gestión de bebi...	Usecase	UC-018	Simple
23	Gestión de bebi...	Usecase	UC-019	Simple

Ilustración 26. Casos de uso por módulo EZEstimate.

Estimation Summary

UAW	10
UUCW	260
UUPC = UAW + UUCW	270
TFactor	33
EFactor	23
TCF = 0.6 + (.01*TFactor)	0,93
EF = 1.4 + (-0.03*EFactor)	0,71
UCP = UUCP*TCT*EF	178,281
Total Effort @ 6,5 Hrs/UCP	1158,8265

Ilustración 27. Estimación del EZEstimate.

6.3 PLANIFICACIÓN TEMPORAL

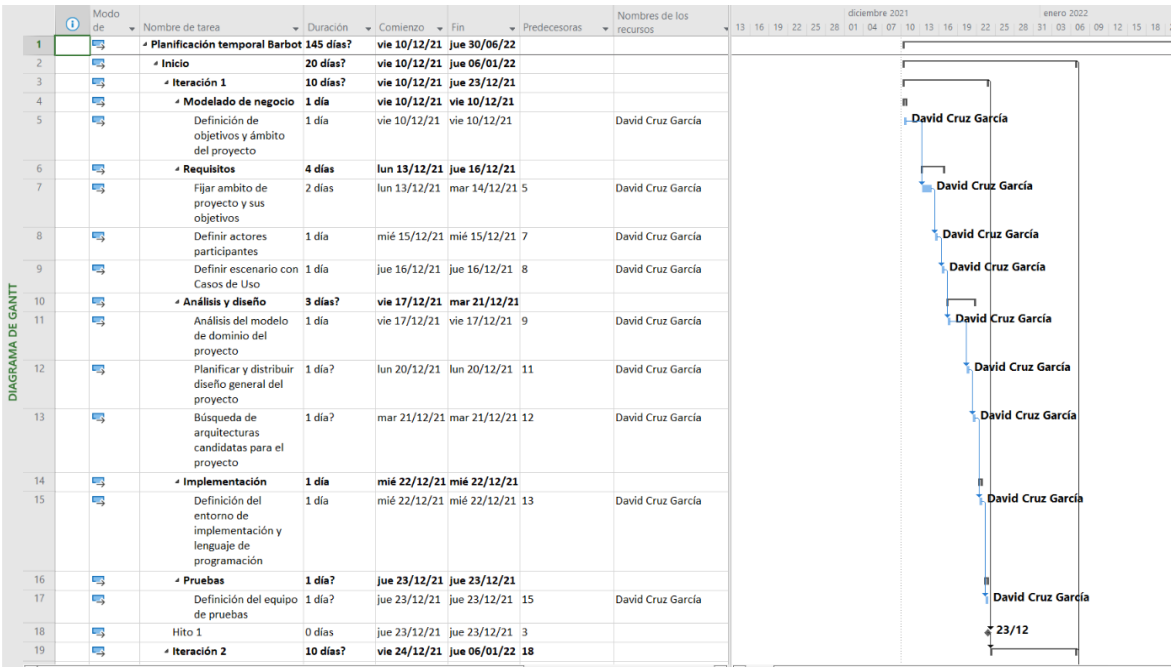
Una vez conocido el tiempo estimado, se procede a realizar la planificación temporal de las diferentes tareas con el objetivo de tener conocimiento sobre cuales son realmente necesarios para el diseño del proyecto.

Para la realización de esta planificación temporal se sigue el esquema del marco de trabajo del Proceso Unificado, llevando a cabo las siguientes estapas:

- ***Modelado de negocio:*** se llevará a cabo una investigación y análisis de aspectos relativos al proyecto a desarrollar tales como investigación y búsqueda de recursos para estar al tanto de aquello que se relaciones con el proyecto a desarrollar para mantenerlo actualizado.
- ***Requisitos:*** se fijarán los objetivos y requisitos que se deben cumplir
- ***Análisis y diseño:*** se realizará un análisis exhaustivo de los requisitos especificados en la etapa anterior, además de determinar el diseño y funcionamiento.
- ***Implementación:*** programación de la aplicación del sistema y construcción del robot.
- ***Pruebas:*** realización de pruebas de parte o todo el sistema para comprobar el correcto funcionamiento y corregir errores de etapas anteriores.

Para obtener más información sobre este apartado, consultar el anexo "*Anexo I. Plan de proyecto".*

En la siguiente figura ("*Ilustración 28")* se pueden observar algunas de las tareas asignadas que se deben llevar a cabo, junto a su diagrama de Gantt:



	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1		**Planificación temporal Barbot**	**145 días?**	**vie 10/12/21**	**jue 30/06/22**		
2		**Inicio**	**20 días?**	**vie 10/12/21**	**jue 06/01/22**		
3		**Iteración 1**	**10 días?**	**vie 10/12/21**	**jue 23/12/21**		
4		**Modelado de negocio**	**1 día**	**vie 10/12/21**	**vie 10/12/21**		
5		Definición de objetivos y ámbito del proyecto	1 día	vie 10/12/21	vie 10/12/21		David Cruz García
6		**Requisitos**	**4 días**	**lun 13/12/21**	**jue 16/12/21**		
7		Fijar ambito de proyecto y sus objetivos	2 días	lun 13/12/21	mar 14/12/21	5	David Cruz García
8		Definir actores participantes	1 día	mié 15/12/21	mié 15/12/21	7	David Cruz García
9		Definir escenario con Casos de Uso	1 día	jue 16/12/21	jue 16/12/21	8	David Cruz García
10		**Análisis y diseño**	**3 días?**	**vie 17/12/21**	**mar 21/12/21**		
11		Análisis del modelo de dominio del proyecto	1 día	vie 17/12/21	vie 17/12/21	9	David Cruz García
12		Planificar y distribuir diseño general del proyecto	1 día?	lun 20/12/21	lun 20/12/21	11	David Cruz García
13		Búsqueda de arquitecturas candidatas para el proyecto	1 día?	mar 21/12/21	mar 21/12/21	12	David Cruz García
14		**Implementación**	**1 día**	**mié 22/12/21**	**mié 22/12/21**		
15		Definición del entorno de implementación y lenguaje de programación	1 día	mié 22/12/21	mié 22/12/21	13	David Cruz García
16		**Pruebas**	**1 día?**	**jue 23/12/21**	**jue 23/12/21**		
17		Definición del equipo de pruebas	1 día?	jue 23/12/21	jue 23/12/21	15	David Cruz García
18		Hito 1	0 días	jue 23/12/21	jue 23/12/21	3	
19		**Iteración 2**	**10 días?**	**vie 24/12/21**	**jue 06/01/22**	**18**	

Ilustración 28. Algunas tareas planificadas junto al diagrama de Gantt.

Como se puede observar, el resultado en días conseguido tras planificar totalmente las tareas es de 145 días, mientras que el resultado estimado fue de 144,75 días. Con esto se puede llegar a la conclusión que dicha planificación es apta para su seguimiento, por lo que se pasa a la siguiente fase.

6.4 ESPECIFICACIÓN DE REQUISISTOS

Una vez finalizada la fase inicial de planificación temporal del sistema se pasa a la fase de elicitación de requisitos software. En esta fase se recogieron todos los requisitos que deberá satisfacer el sistema. Para ello, se utilizó la metodología Durán y Bernárdez para elicitación de requisitos de un sistema software.

Para obtener más información sobre este apartado, consultar el anexo "*Anexo II. Especificación de requisitos del sistema".*

A continuación, se mostrarán a modo de ejemplo algunos casos de eso según cada etapa de esta fase de especificación de requisitos. El resto se podrán consultar en el anexo mencionado previamente.

6.4.1 PARTICIPANTES

El proyecto cuenta con varios participantes: el alumno o autor y los tutores encargados del proyecto:

- ***Cruz García, David:*** alumno y autor del proyecto
- ***Sales Mendes, André Filipe***
- ***Villarrubia González, Gabriel***
- ***De Paz Santana, Juan Francisco***

Estos participantes se recogerán en sus respectivas tablas junto a la tabla de la organización a la que pertenecen ("*Universidad de Salamanca"*).

6.4.2 OBJETIVOS DEL SISTEMA

Se han definido varios objetivos del sistema que se deberán cumplir en la finalización del desarrollo de este proyecto:

- ***Preparación de bebidas o cocteles***
- ***Gestión del brazo robótico***
- ***Altas, bajas y modificaciones de datos de usuarios***
- ***Altas, bajas y modificaciones de datos de bebidas***
- ***Altas, bajas y modificaciones de datos de recetas***
- ***Personalización en la disposición de las bebidas en el robot***
- ***Almacenamiento y visualización de estadísticas***

Al igual que ocurre con los participantes, estos objetivos se recogen en sus tablas correspondientes. Se adjunta una de estas tablas como ejemplo ("*Tabla 1")*:

OBJ-001	Preparación de bebidas o coctels
Versión	1.0
Autores	*Cruz García, David*
Fuentes	*Sales Mendes, André Filipe* *Villarrubia González, Gabriel* *De Paz Santana, Juan Francisco*
Descripción	El sistema deberá *ser capaz de preparar las bebidas según la receta indicada.*
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediato
Estado	Validado
Estabilidad	Alta
Comentarios	Sin comentarios

Tabla 1. OBJ-001. Preparación de bebidas o coctels.

6.4.3 REQUISITOS DE INFORMACIÓN

Estos requisitos indican los datos que almacenará y gestionará el sistema para su correcto funcionamiento. Los requisitos de información especificados para el desarrollo de este proyecto son:

- ***Información sobre usuarios***
- ***Información sobre roles***
- ***Información sobre roles que corresponden a cada usuario***
- ***Información sobre las bebidas***
- ***Información sobre las posiciones de las bebidas***
- ***Información sobre las recetas***
- ***Información sobre las bebidas de las recetas***
- ***Información sobre las estadísticas***

Se adjunta como ejemplo una tabla (*"Tabla 2"*) donde se recoger uno de estos requisitos de información:

IRQ-006	Información sobre las recetas
Versión	1.0
Autores	*Cruz García, David*
Fuentes	*Sales Mendes, André Filipe* *Villarrubia González, Gabriel* *De Paz Santana, Juan Francisco*
Dependencias	*OBJ-001. Preparación de bebidas o coctels.* *OBJ-005. Altas, bajas y modificación de datos de recetas.* OBJ-006. Personalización en la disposición de las bebidas en el robot.
Descripción	El sistema deberá *almacenar información sobre las recetas añadidas creadas por el administrador*
Datos específicos	- *ID* - *Nombre* - *Descripción* - *Imagen* - *Activa*
Importancia	Vital
Urgencia	Inmediato
Estado	Validado
Estabilidad	Alta
Comentarios	Sin comentarios

Tabla 2. IRQ-006. Información sobre las recetas.

.4.4 REQUISITOS FUNCIONALES

Estos requisitos definen el comportamiento del sistema según la situación en la que se encuentra. Para la especificación de estos requisitos se divide el sistema en varios paquetes que incluirán casos de usos relacionados con un propósito. En el caso de este proyecto, habrá 4 paquetes haciendo referencia a los 4 pilares principales del sistema: los usuarios, las bebidas, las recetas y el robot (ver "*Ilustración 29")*

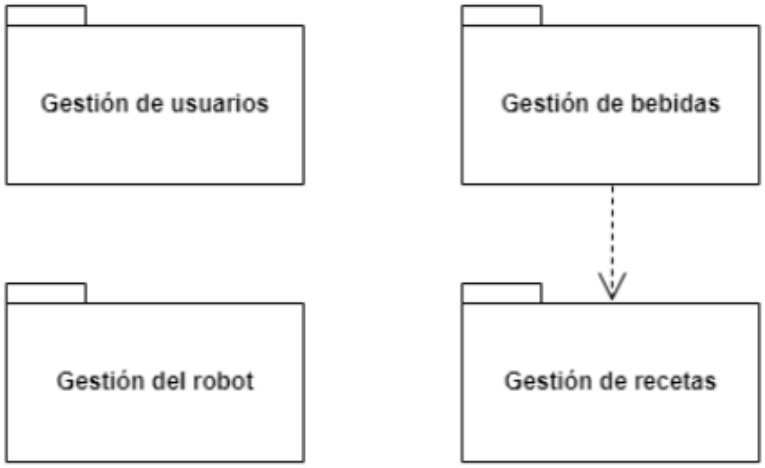


Ilustración 29. Diagrama de paquetes de requisitos funcionales.

Una vez dividido el sistema en paquetes, se definen los actores que interactuarán con el sistema. Estos son:

- **Usuario no logueado:** usuario que puede solicitar la preparación de bebidas sin necesidad de registro y que puede acceder al sistema realizando login con credenciales de usuario.
- **Usuario logueado:** usuario que ha accedido al sistema mediante unas credenciales y que puede realizar acciones adicionales a la solicitud de bebidas.
- **Usuario cliente:** especialización del usuario logueado que hace referencia a un cliente que puede solicitar bebidas y que dispone de un perfil y unas estadísticas asociadas. Dentro del sistema tendrá un rol "cliente"
- **Usuario administrador:** especialización del usuario logueado que hace referencia al administrador completo del sistema. Es el encargado de crear las bebidas, recetas, gestionar la aplicación y gestionar el robot. Dentro del sistema tendrá un rol de "admin"
- **Sistema:** usuario que hace referencia al propio sistema. El sistema actuará como un "usuario" en ciertas ocasiones como en el cálculo de estadísticas donde realizará ciertas acciones automáticas.

Estos actores se organizan siguiendo una jerarquía de actores, por lo que a continuación se mostrará esta jerarquía junto con una tabla a modo de ejemplo (*"Ilustración 30"*):

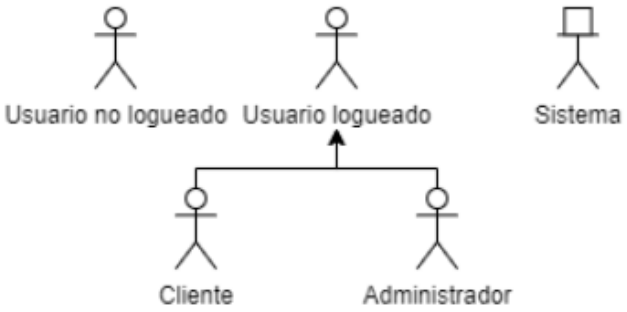


Ilustración 30. Jerarquía de actores.

Finalmente, se definen los distintos casos de uso del sistema, organizados todos ellos por paquetes en un diagrama de casos de uso para ese paquete. Se incluye un diagrama de casos de uso (*"Ilustración 31"*) y una tabla de caso de uso a modo de ejemplo de esta de recopilación de requisitos funcionales ("*Tabla 3"*).

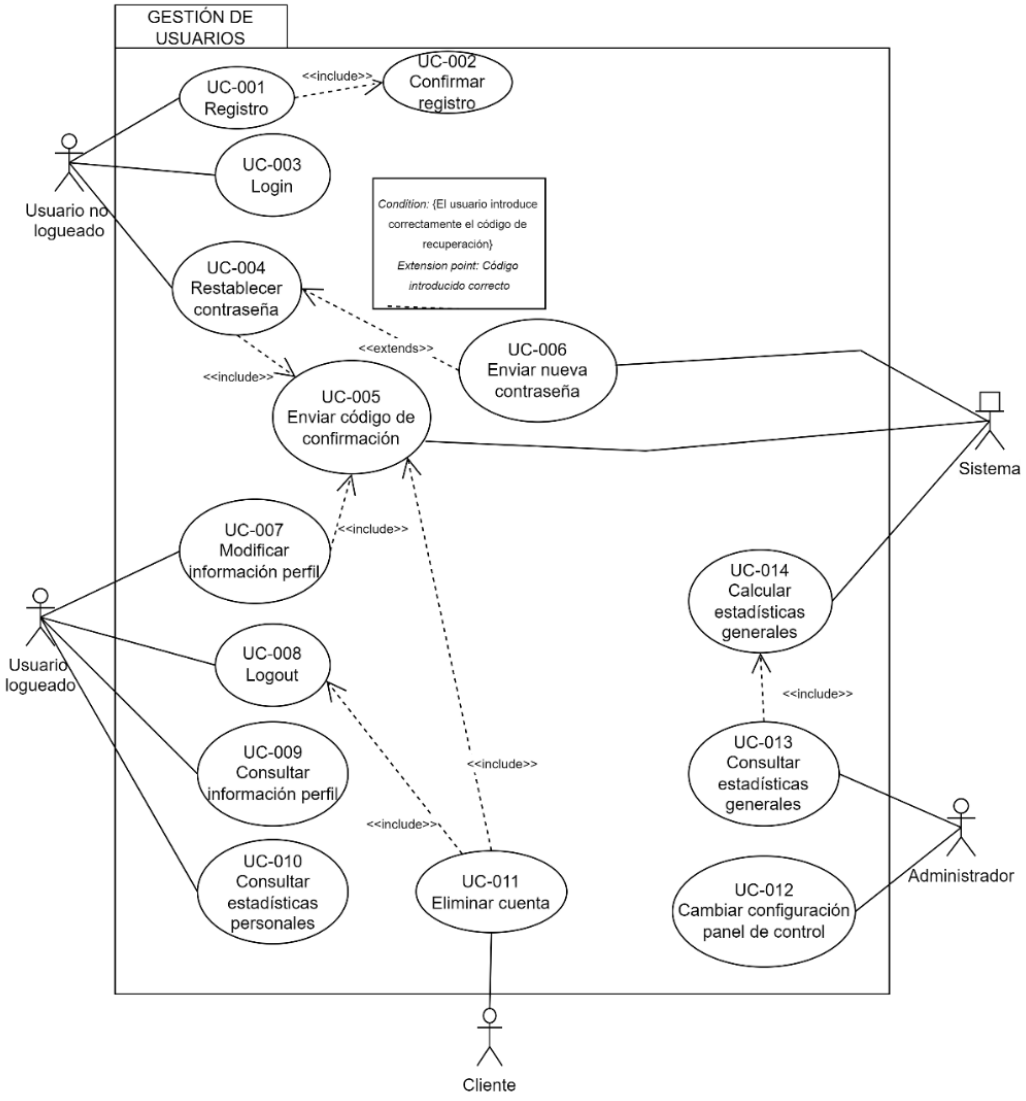


Ilustración 31. Jerarquía CU de gestión de usuarios.

UC-015	**Añadir bebida**	
Versión	1.0	
Autores	*Cruz García, David*	
Fuentes	*Sales Mendes, André Filipe* *Villarrubia González, Gabriel* *De Paz Santana, Juan Francisco*	
Dependencias	*OBJ-001. Preparación de bebidas o coctels.* *OBJ-004. Altas, bajas y modificación de bebidas.* *OBJ-005. Altas, bajas y modificación de datos de recetas.* *OBJ-006. Personalización en la disposición de las bebidas en el robot.*	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor "ACT-004 Administrador" pretenda añadir una nueva bebida en el sistema.*	
Precondición	El actor "*ACT-004 Administrador*" ha iniciado sesión en el sistema y tiene una sesión activa.	
Secuencia normal	**Paso**	**Acción**
	p_1	El actor "*ACT-004*" solicita añadir una bebida en el sistema.
	p_2	El sistema solicita los datos de la bebida al actor "*ACT-001*".
	p_3	El actor "*ACT-001*" proporciona los datos solicitados.
	p_4	El sistema valida los datos introducidos.
	p_5	El sistema almacena convenientemente los datos introducidos
Postcondición	El actor se queda en estado no valido a la espera de la verificación.	
Excepciones	**Paso**	**Acción**
	p_4	Si la validación de datos falla, el sistema informa al usuario y se regresa al paso p_2
	p_5	Si la bebida ya está dentro del sistema, u ocurre algún tipo de error, el sistema informará al usuario, y el caso de uso queda sin efecto.
Importancia	Vital	
Urgencia	Inmediato	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Sin comentarios	

Tabla 3. UC-015. Añadir bebida.

6.4.5 REQUISITOS NO FUNCIONALES

Finalmente se especifican los requisitos no funcionales del sistema. Estos requisitos especifican criterios o restricciones del sistema que afectan a su diseño e implementación.

Los requisitos no funcionales planteados son los siguientes:

- ***Portabilidad***
- ***Manejabilidad***
- ***Usabilidad***
- ***Confiabilidad***
- ***Disponibilidad***
- ***Despliegue***

6.4.6 MATRIZ DE RASTREABILIDAD

Una vez se han especificado todos los requisitos del sistema, se crea la matriz de rastreabilidad, que permite ver rápidamente y de forma visual la relación entre cada requisito definido con cada objetivo a cumplir del proyecto.

6.5 ANÁLISIS DE REQUISITOS

Finalizada la fase de elicitación de requisitos pasamos a la fase de análisis de estos mismos.

Para obtener más información sobre este apartado, consultar el anexo "*Anexo III. Análisis de requisitos".*

6.5.1 MODELO DE DOMINIO

Es un modelo que recoge una serie de clases que representan el mundo real, recogiendo así las necesidades de almacenamiento que tiene el sistema y cómo el sistema trabaja y gestiona esta información.

Para ello se emplea un diagrama de clases. En este diagrama se recogen las clases (entidades del modelo de negocio), sus atributos y las relaciones entre ellas, representando así la información con la que trabajará el sistema.

El diagrama de clases empleado para el modelo de dominio de este sistema ha sido el siguiente (*"Ilustración 32"*):

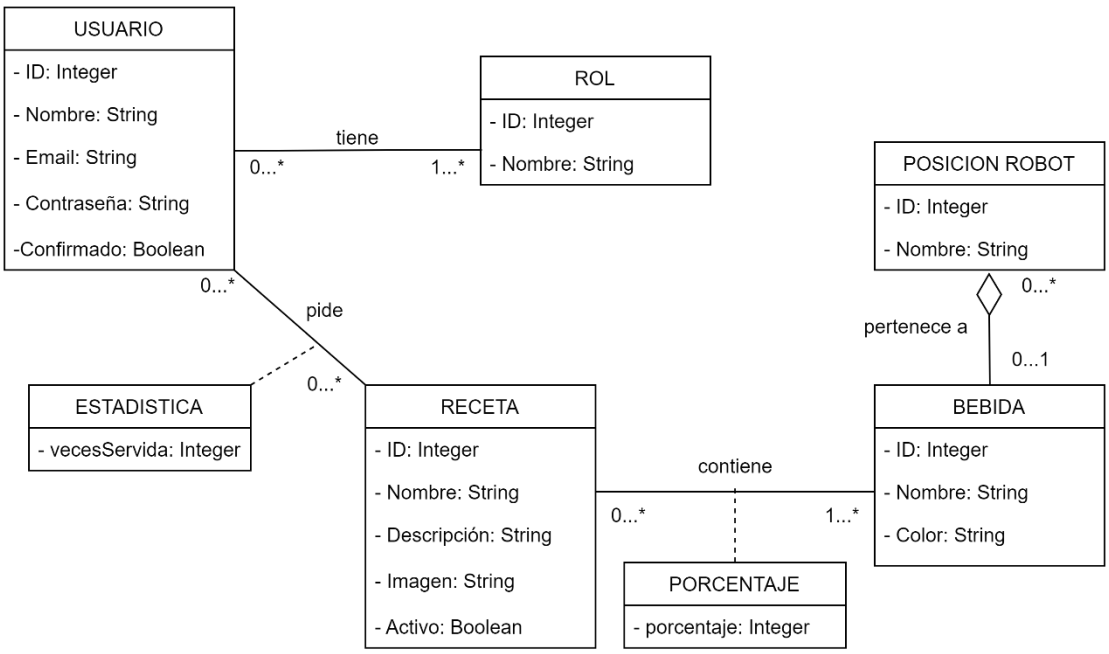


Ilustración 32. Modelo de dominio. Diagrama de clases.

6.5.2 REALIZACIÓN DE CASOS DE USO (ANÁLISIS)

Para mostrar dichas interacciones entre los distintos elementos del sistema se hará a través de los diagramas de secuencia asociados a cada caso de uso.

A modo de ejemplo, se adjunta uno de estos diagramas de secuencia (*"Ilustración 33)*:

UC-024	**Asignar bebidas a receta**

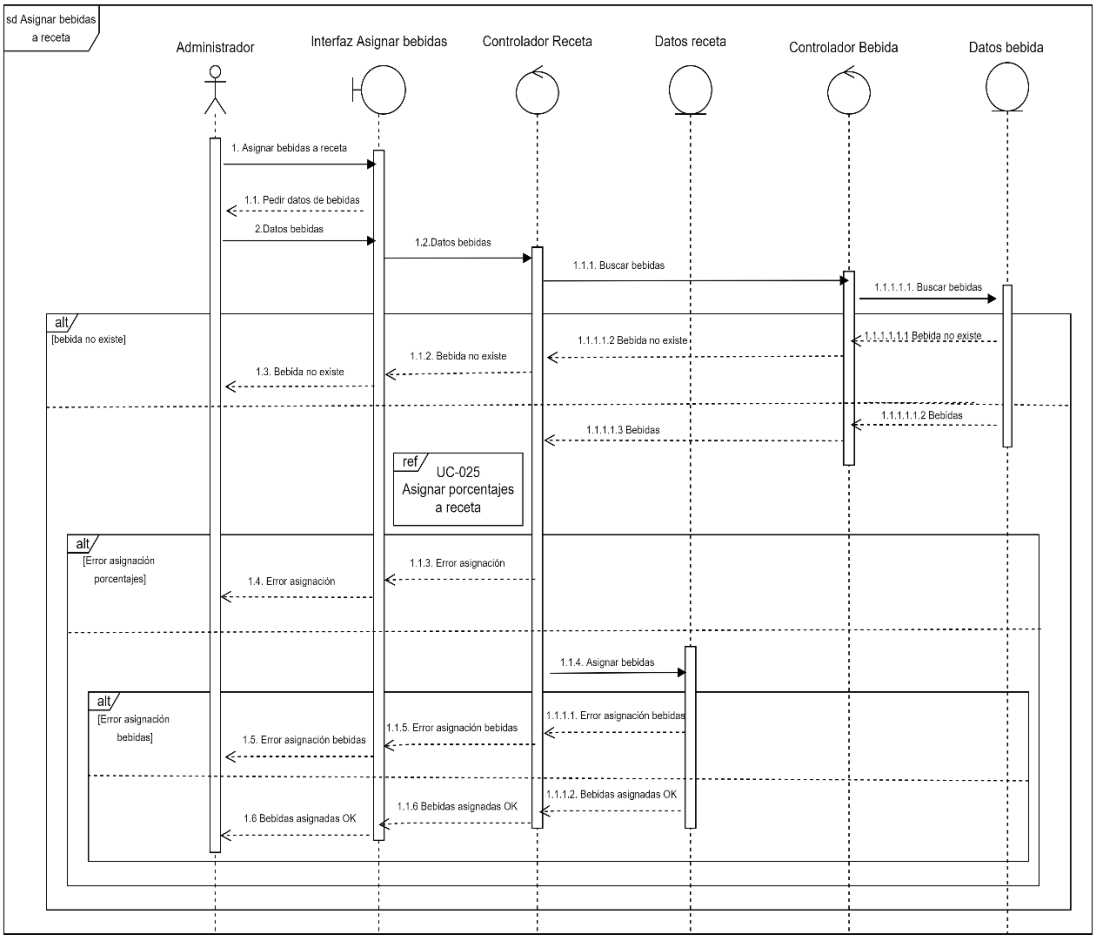


Ilustración 33. Diagrama de secuencia UC-024. Asignar bebidas a receta.

6.5.3 CLASES DE ANÁLISIS

Una vez se han diseñado los diagramas de secuencia mostrando el paso de mensajes entre distintos los distintos elementos del sistema para llevar a cabo los requisitos planteados, se pasa a explicar los diagramas de comunicación, también divididos por paquetes, donde se puede observar la comunicación entre los distintos componentes (interfaces, controladores y datos) del sistema planteados en los diagramas de secuencia.

A modo de ejemplo, se adjunta uno de estos diagramas de comunicación (*"Ilustración 34"*):

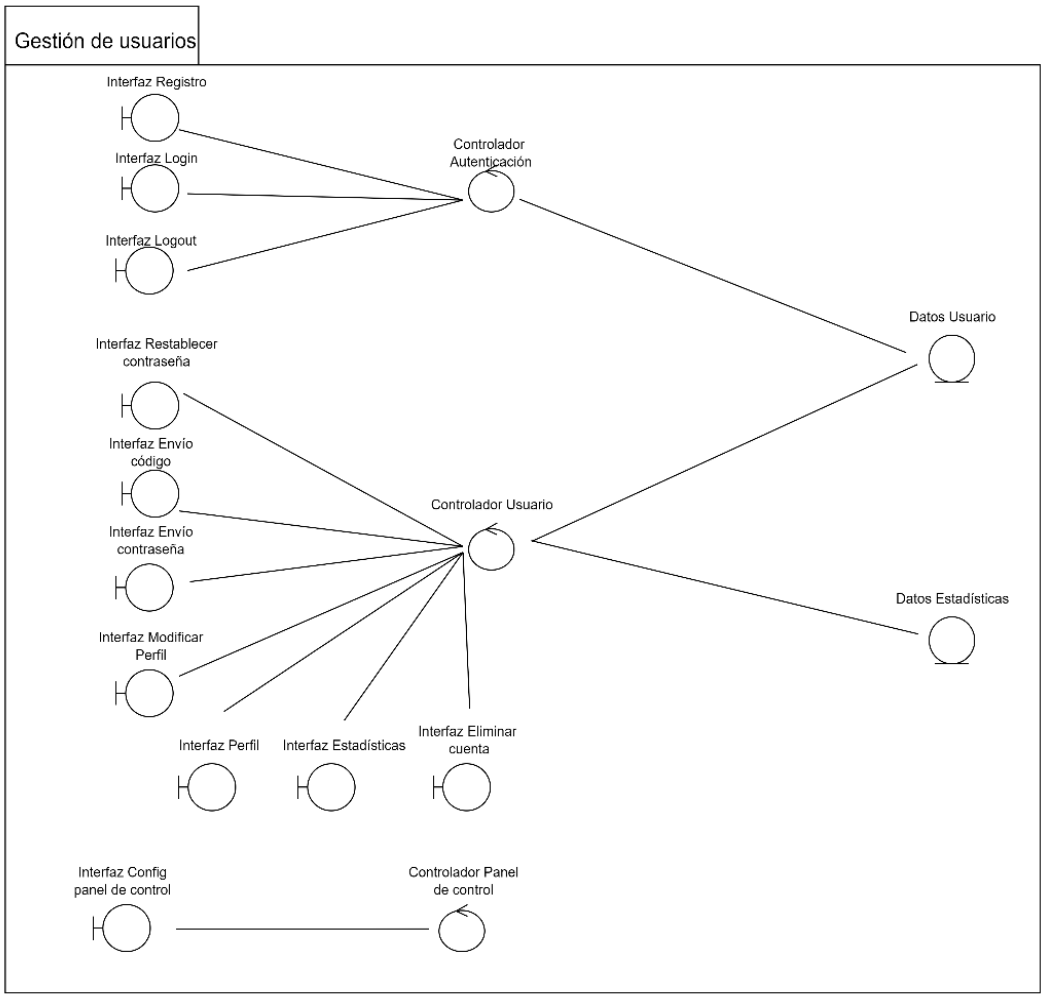


Ilustración 34. Diagrama comunicación paquete Gestión de Usuarios.

6.5.4 VISTA ARQUITECTÓNICA DEL MODELO DE ANÁLISIS

La vista arquitectónica del sistema permite situar cada clase de análisis distribuida en el apartado anterior dentro del patrón arquitectónico *"MVC: Modelo-Vista-Controlador"*. Consultar el documento "*Anexo III – Análisis de requisitos"* para más información.

6.6 DISEÑO DEL SISTEMA

Una vez definidos todos los casos de uso del sistema y analizados presentando el modelo de dominio del sistema, hay que comenzar a diseñar el sistema, tanto su estructura mediante el uso de patrones como las diferentes clases de diseño, preparando el proyecto para su implementación.

Para obtener más información sobre este apartado, consultar el anexo "*Anexo IV. Diseño del sistema software".*

6.6.1 PATRONES DE ARQUITECTURA

6.6.1.1 PATRÓN BFF

El patrón ***BFF o Backend-For-Frontend*** es un patrón de diseño software enfocado a servicios REST, enfocando el desarrollo del backend totalmente a la estructuara del frontend. Se puede observar este patrón en la *"Ilustración 35".*

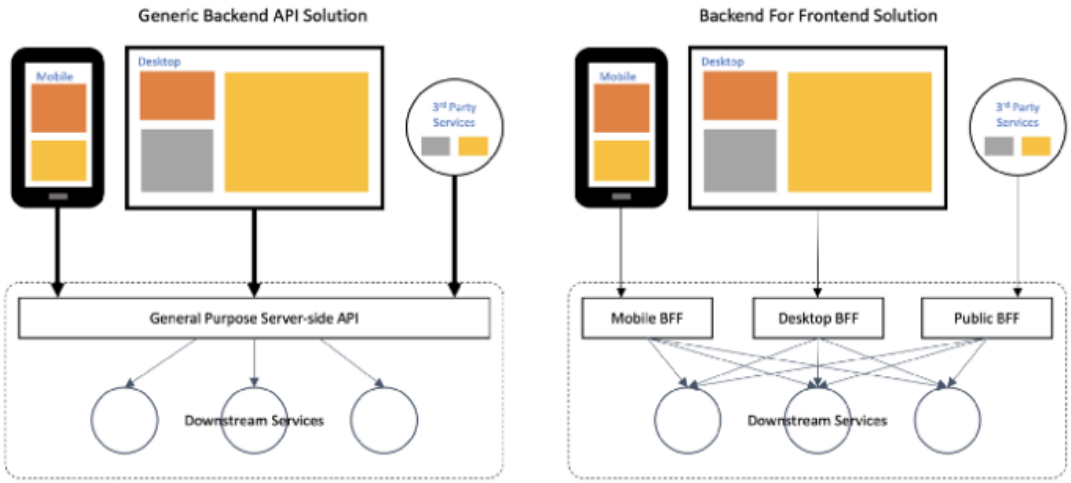


Ilustración 35.. Patrón BFF.

6.6.1.2 PATRÓN ACCESS TOKEN

El patrón ***Access Token*** es un patrón arquitectónico que sirve para autenticar, utilizando para ello un token de acceso (en mi caso, JWT), las peticiones enviadas. Se puede observar el modo de trabajar con tokens en la "*Ilustración 36".*

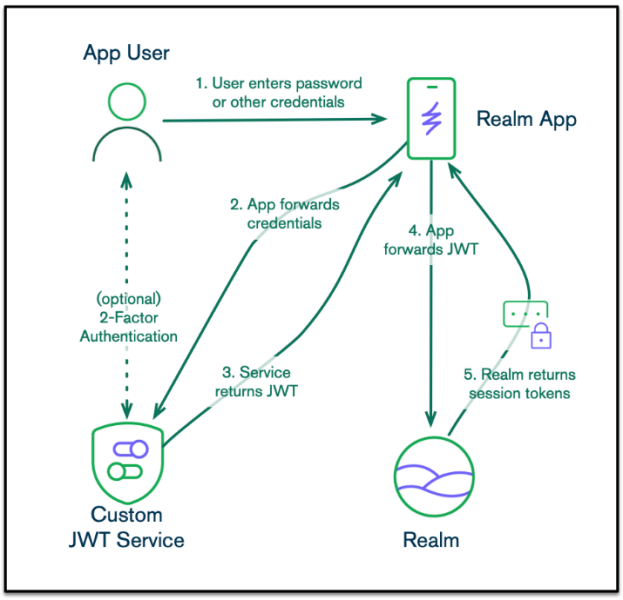


Ilustración 36. Patrón Access Token con JWT

6.6.1.3 PATRÓN FACADE

El patrón ***Facade o Fachada*** es un patrón que oculta la complejidad para acceder a un determinado objeto o servicio, proporcionando para ello una interfaz de más alto nivel. Se puede observar su funcionamiento en la "*Ilustración 37".*

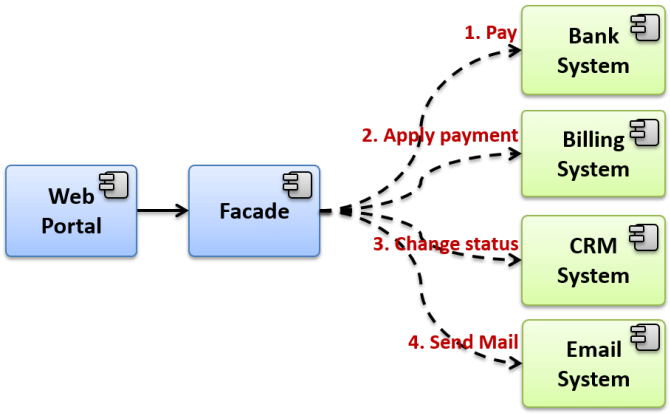


Ilustración 37. Patrón Facade.

6.6.2 SUBSISTEMAS DE DISEÑO

Una vez superada la etapa de análisis del proyecto y establecidos los patrones arquitectónicos que se utilizarán durante el diseño, se realiza una división del sistema en subpaquetes más específicos dentro de la estructura general de la aplicación. La descripción de estos paquetes puede ser consultado en el anexo *"Anexo IV – Diseño del sistema software".* Se pueden observar estos subsitemas de diseño en la "*Ilustración 38".*

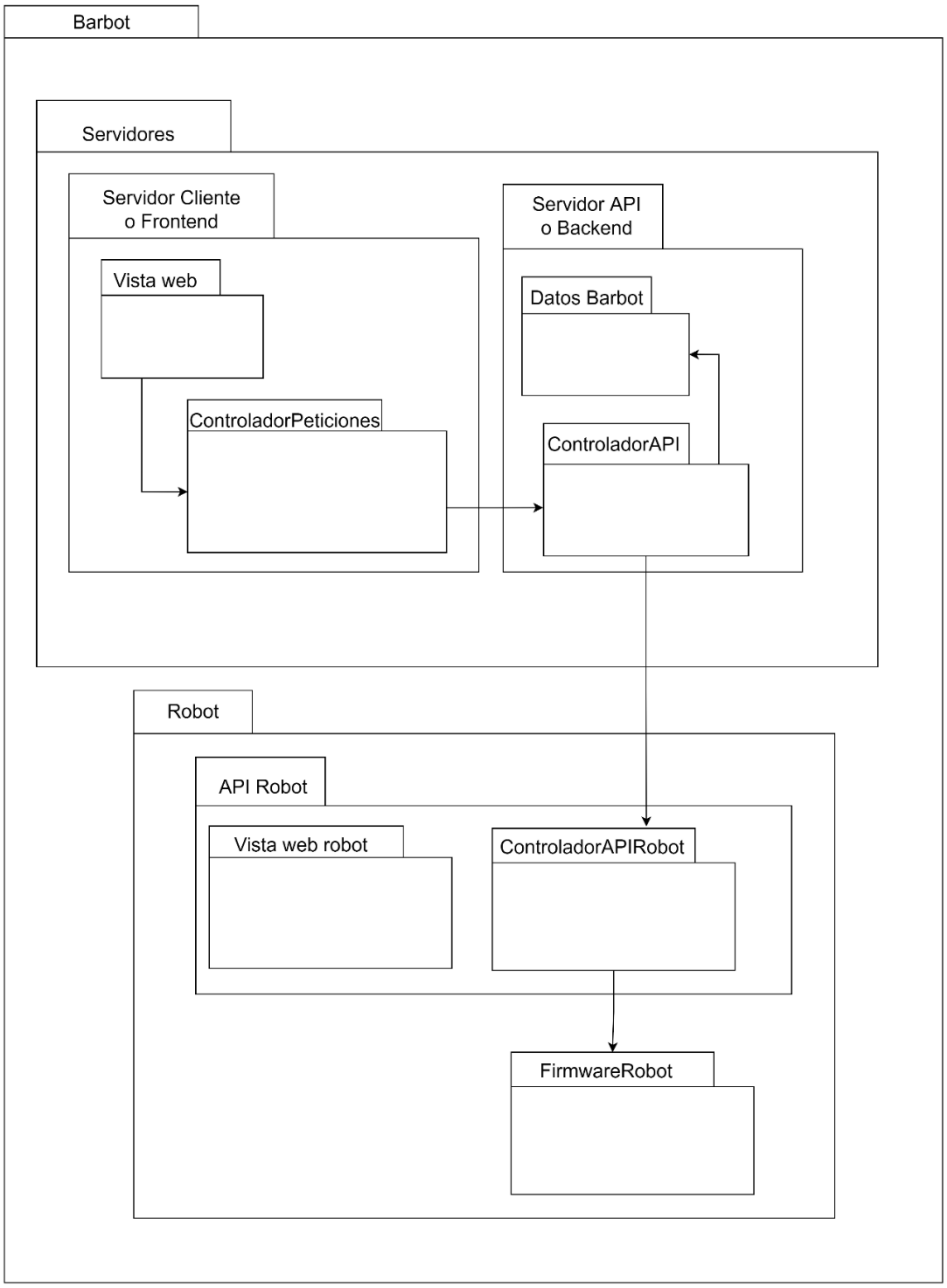


Ilustración 38. Subsistemas de diseño.

6.6.3 CLASES DE DISEÑO

Este apartado recopilará los contenidos del sistema en clases con sus respectivos métodos (clases de diseño). Se adjuntarán 2 diagramas de diseño correspondientes a los controladores de los servidores a modo de ejemplo. El primer controlador se corresponde al encargado en el servidor web de realizar las peticiones a la API, sirviendo de fachada (patrón Facade) a las distintas vistas web (“*Ilustración 39”)*. El segundo controlador corresponde al controlador del lado del servidor API, que recibirá las peticiones del primer controlador (el del servidor web), realizará distintas acciones y devolverá los resultados (“*Ilustración 40”)*.

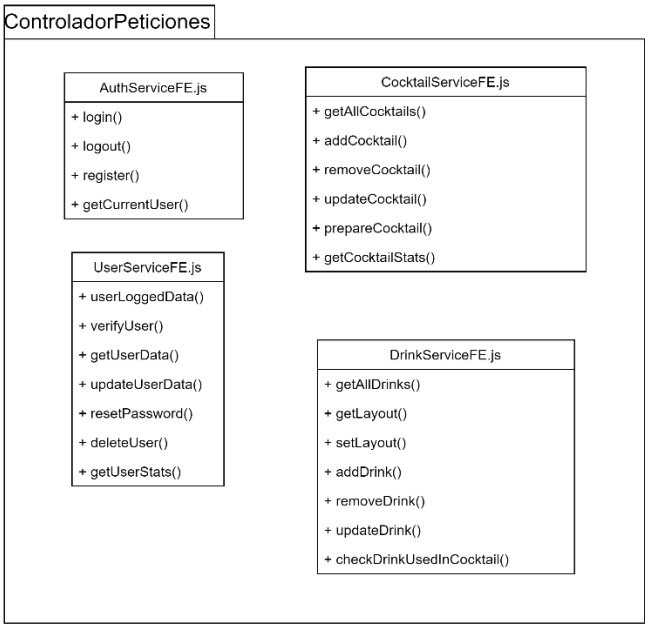


Ilustración 39. Controlador de peticiones servidor web.

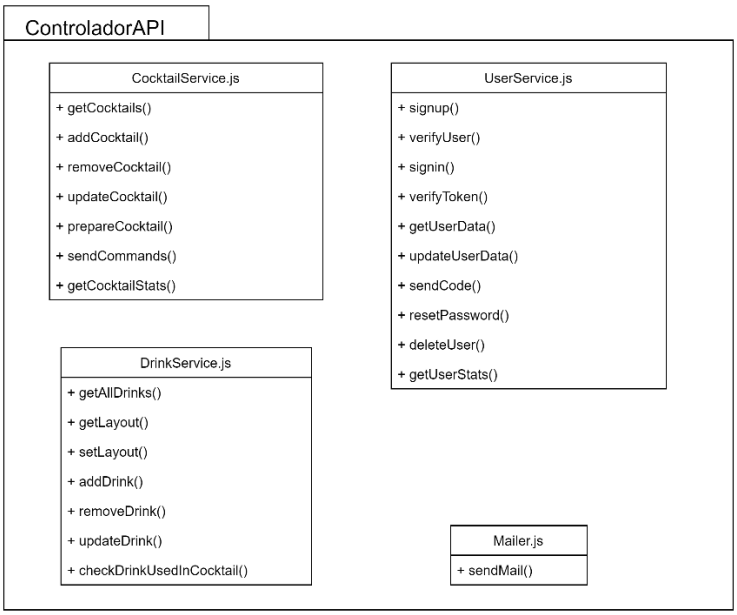


Ilustración 40. Controlador API.

6.6.4 VISTA ARQUITECTÓNICA DEL MODELO DE DISEÑO

Gracias a la vista arquitectónica, se da una imagen de la implementación del patrón arquitectónico BFF (*Backend For Frontend)* en la estructura del sistema diseñado. Además, se pueden dividir, al igual que en la etapa de análisis, las clases de diseño según el patrón MVC (Modelo-Vista-Controlador), dando una visión general del sistema separada por las vistas, los controladores y los datos almacenados (modelo).

Todo esto se puede observar en la "*Ilustración 41".*

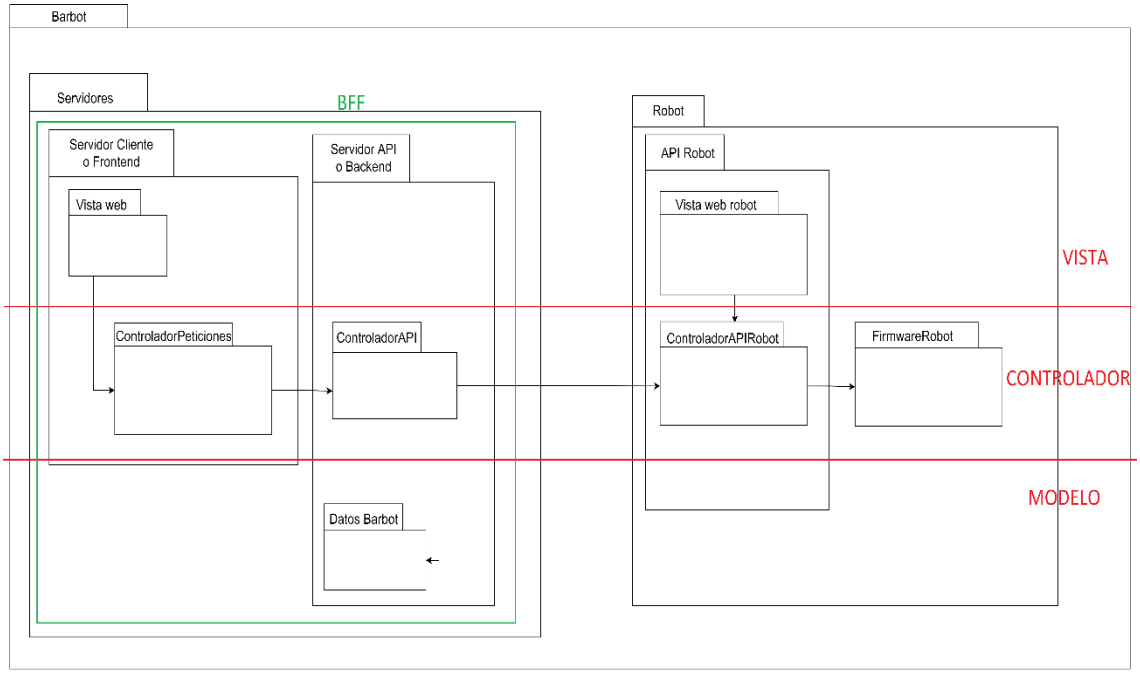


Ilustración 41. Vista arquitectónica (diseño).

6.6.5 REALIZACIÓN DE CASOS DE USO (DISEÑO)

El objetivo de este apartado es detallar el intercambio de mensajes entre objetos mediante diagramas de secuencia de diseño, al igual que se hizo en la etapa de análisis, pero esta vez junto a las clases de diseño creadas. Para ello se sigue la misma división en paquetes realizada durante la etapa de análisis.

Se adjunta a modo de ejemplo un diagrama de secuencia de diseño ("*Ilustración 42")*:

UC-015	**Añadir bebida**

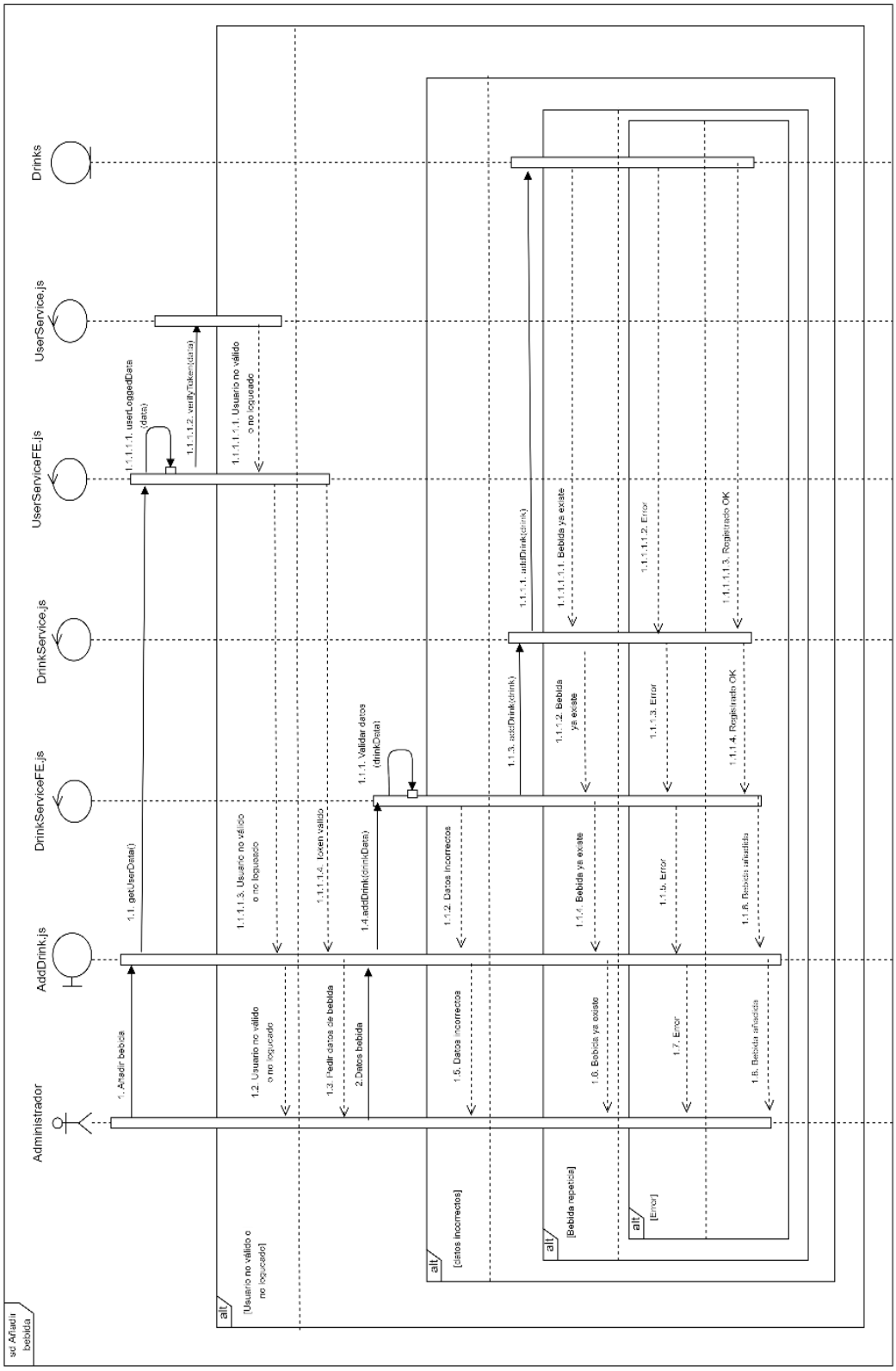


Ilustración 42. Diagrama de secuencia UC-015 (diseño).

6.6.6. DISEÑO DE LA BASE DE DATOS

Para que el sistema pueda almacenar y gestionar correctamente la información, se utilizará una base de datos MySQL. En esta base de datos se guardarán diferentes tablas correspondientes a los datos de los usuarios, bebidas, recetas y otros de interés para el sistema.

Esta base de datos se describe según el siguiente modelo de entidad relación (“*Ilustración 43”*):

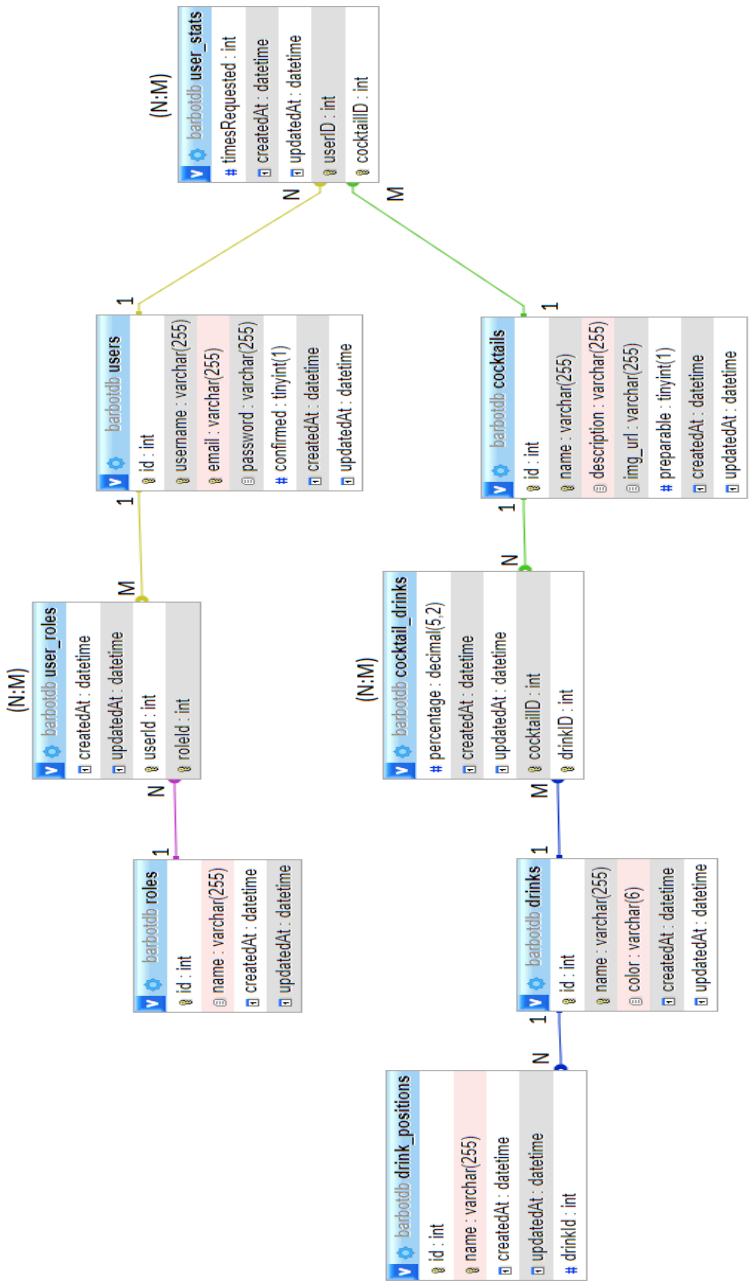


Ilustración 43. Diseño base de datos.

6.6.7. MODELO DE DESPLIEGUE

El modelo de despliegue se representa utilizando para ello un diagrama de despliegue. Este diagrama muestra la arquitectura de ejecución de un sistema, incluyendo para ellos entornos de ejecución hardware, software y los middlewares que los conectan (“*Ilustración* 44”).

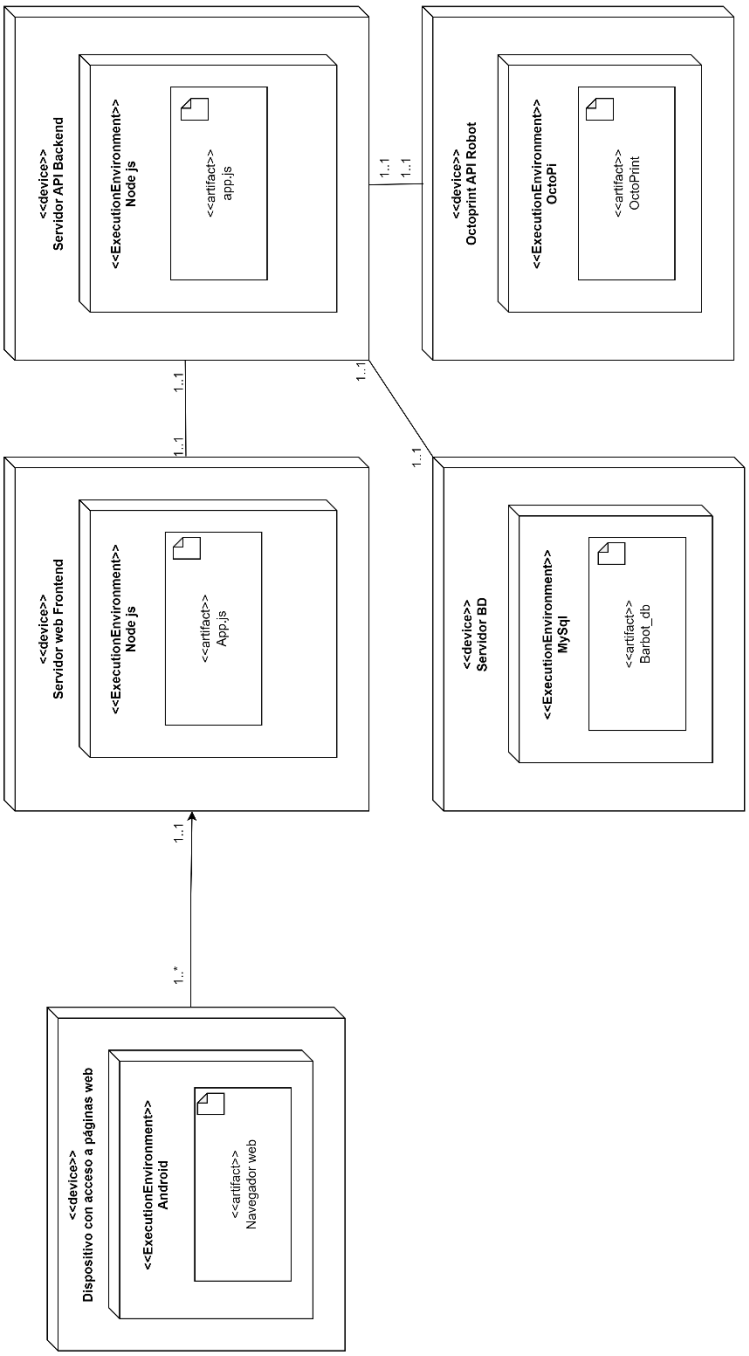


Ilustración 44. Diagrama de despliegue.

6.7 IMPLEMENTACIÓN

Durante esta fase se ha realizado la codificación de todo el sistema tomando los resultados obtenidos en la fase de diseño como base, utilizando para ello las herramientas y técnicas explicadas en el apartado "*5 Herramientas y técnicas empleadas".*

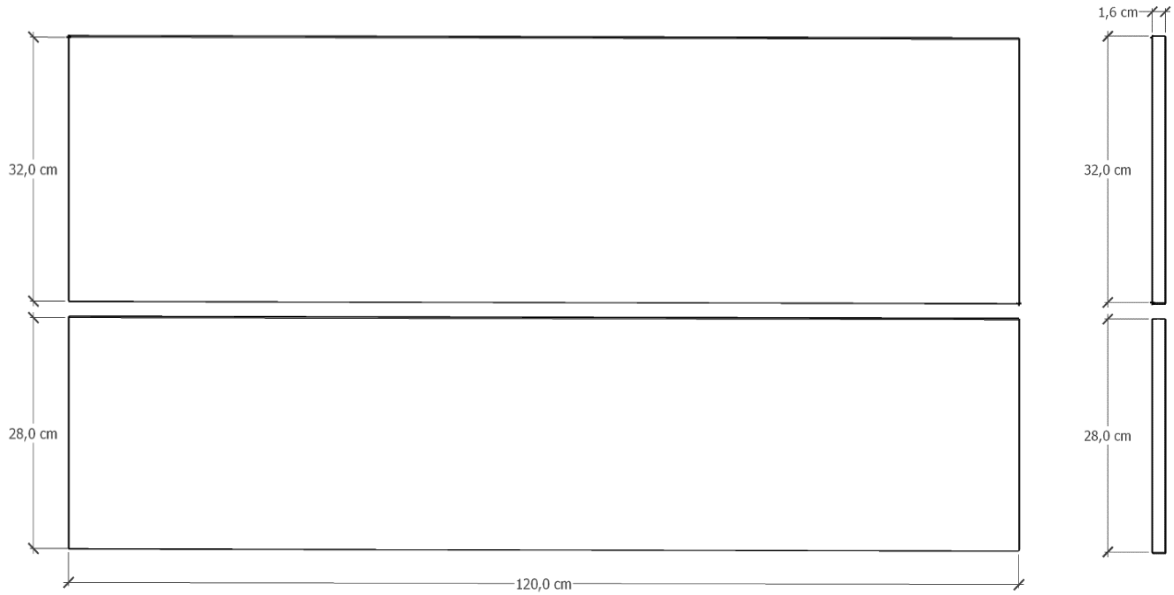
La implementación puede ser dividida en 4 secciones:

- ***Servidor web o cliente:*** codificación de la página web a la que accederán los usuarios para realizar diferentes acciones en el sistema y que se encargará de comunicarse con el servidor API para poder realizar toda la funcionalidad diseñada y mencionada.
- ***Servidor API o servidor:*** codificación de la API REST a la que se comunicará el servidor web para realizar las distintas funcionalidades dentro del sistema. Este servidor será el encargado de comunicarse con el robot, acceder a la base de datos, enviar emails, etc.
- ***Construcción del robot:*** preparación y montaje de la estructura del robot y del propio robot, y diseño y montaje de los esquemas electrónicos a utilizar para el correcto funcionamiento del robot. Este proceso se explicará en el apartado siguiente.
- ***Preparación y codificación del robot:*** implementación de toda la funcionalidad necesaria para poder mover y gestionar el robot de forma correcta desde la aplicación web desarrolla y la API REST que controla todo el robot. Todo lo utilizado para ello está explicado en el apartado 5.

6.8 CONTRUCCIÓN Y ESQUEMAS ELECTRÓNICOS

En este apartado se explicará el proceso de diseño y construcción del robot y de su cableado para su correcto funcionamiento.

En primer lugar, había que diseñar la estructura del robot siguiendo unas medidas que fueran óptimas para la correcta colocación de los dispensadores. Para ello, utilizando máquinas de carpintería, se realizaron varios cortes en un tablón de madera tras diseñar las piezas y obtener sus medidas finales en el programa SketchUp. Se muestran en las figuras "*Ilustración 45",* "*Ilustración 46",* "*Ilustración 47" y* "*Ilustración 48".*



lustración 45. Corte base y fondo del robot.

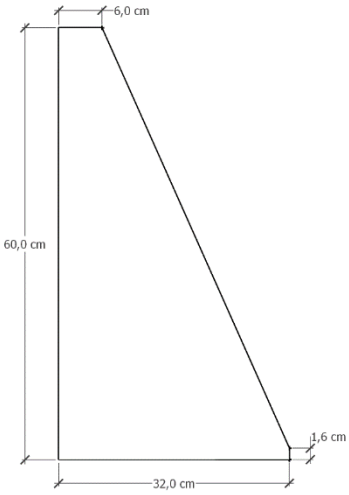


Ilustración 47. Cortes piezas laterales.

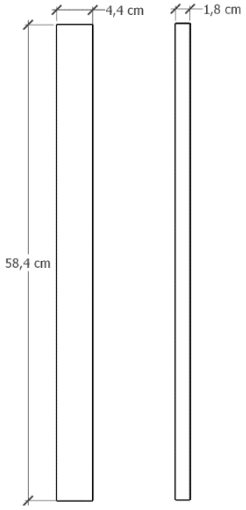


Ilustración 46. Cortes listones.

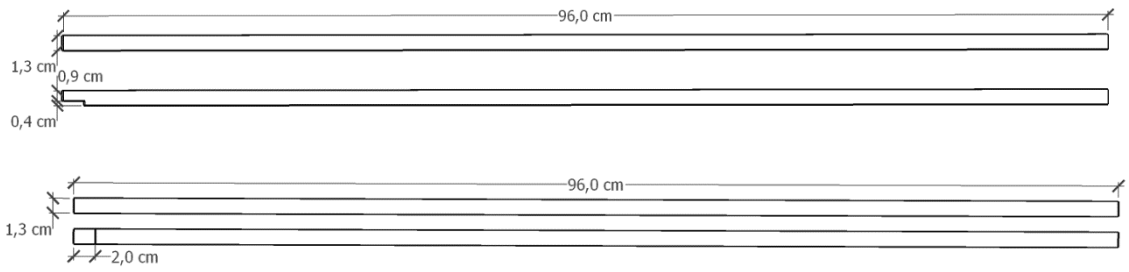


Ilustración 48. Cortes barras de apoyo.

Una vez preparadas las piezas que conforman la estructura, se lleva a cabo su montaje. Este montaje está explicado paso a paso junto con los materiales necesarios en el anexo "*Anexo VII. Esquemas electrónicos, diseño y montaje del robot*", en el apartado "*2 Manual de montaje".*

Ahora, se mostrará únicamente el inicio ("*Ilustración 49")* y el final de la construcción del robot ("*Ilustración 50")*:

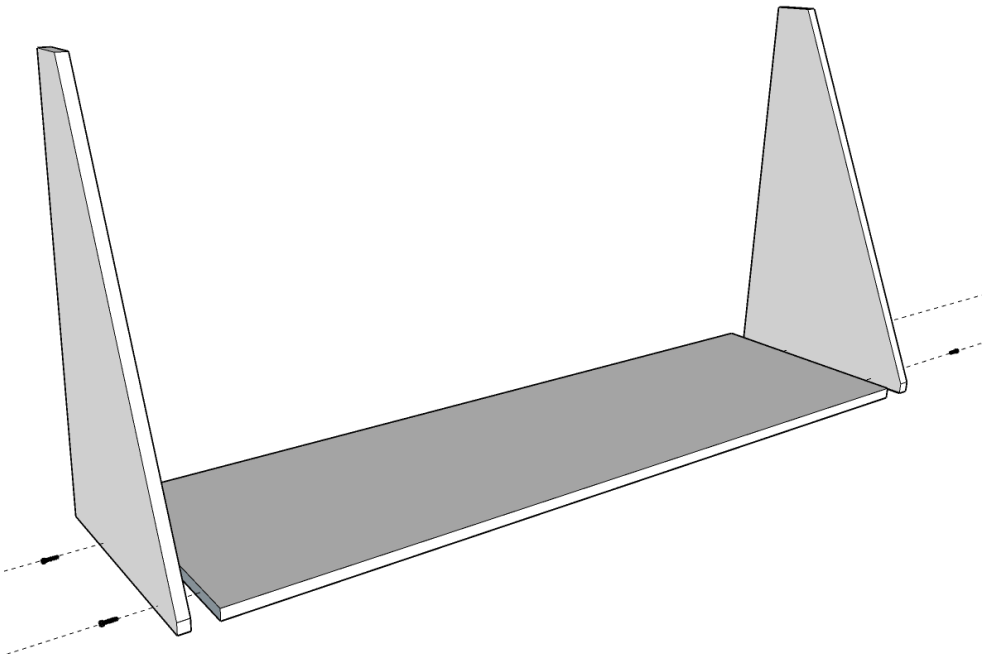


Ilustración 49. Inicio construcción robot.

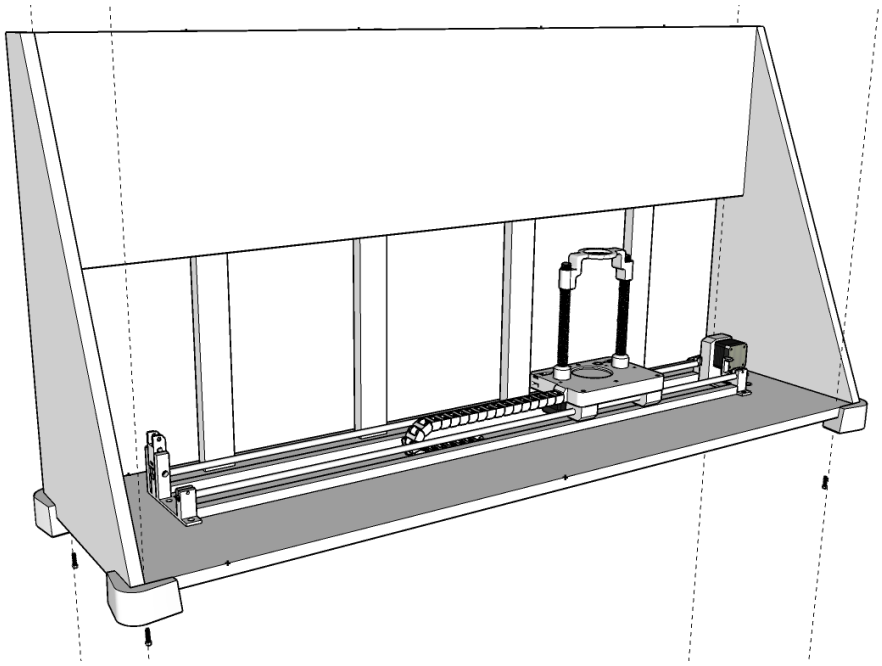


Ilustración 50. Fin construcción robot.

Finalmente, se diseñó el cableado necesario para el funcionamiento del robot, desde el movimiento de los motores, el LED de aviso y el Arduino para el funcionamiento del robot. El cableado completo se muestra en el diagrama de la "*Ilustración 51".*

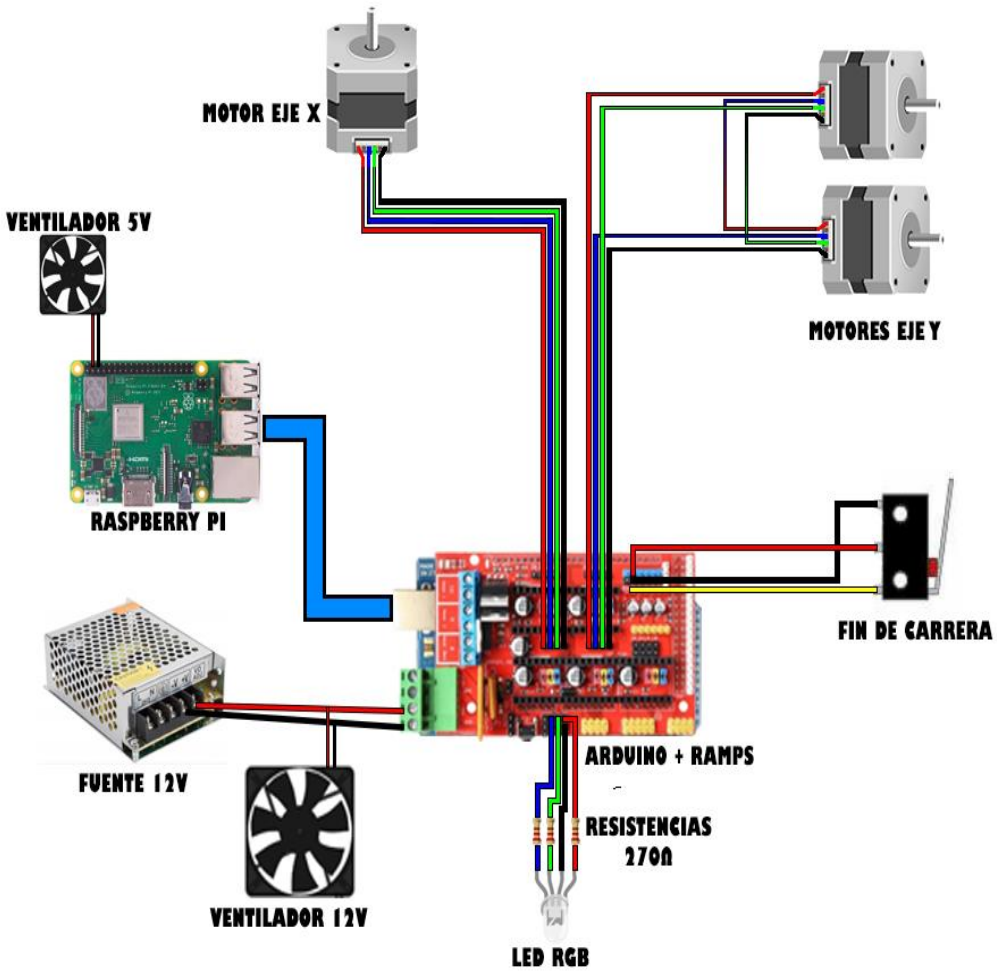


Ilustración 51. Cableado del robot.

Finalmente, podemos ver el resultado final de la construcción de este robot en la siguiente imagen, las figuras "*Ilustración 52"* y *"Ilustración 53"*.

Ilustración 52. Resultado final parte delantera.

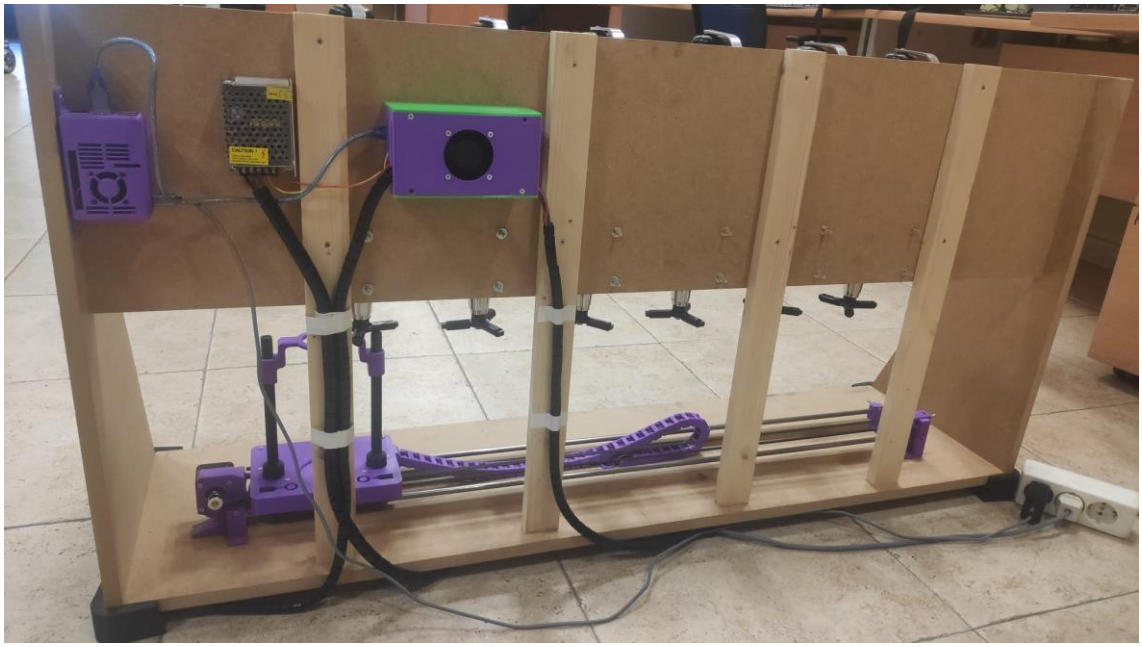


Ilustración 53. Resultado final parte trasera.

6.9 PRUEBAS

Durante todo el desarrollo, tanto de la aplicación como del robot, se ha realizado un número considerable de pruebas con el objetivo de comprobar el correcto funcionamiento del sistema, tanto a nivel de programación como de hardware.

Esta parte ha sido fundamental para detectar pequeños errores que han ido surgiendo a lo largo del desarrollo del proyecto, obteniendo un feedback positivo de lo sucedido e implementando mejores, tanto en el diseño de la aplicación y servidores como en el diseño del robot.

Finalmente, una vez concluido toda la implementación del proyecto, se han realizado pruebas del sistema global asegurando que todos los componentes del mismo funcionen correctamente.

6.10 FUNCIONALIDAD DEL SISTEMA

En este apartado se mostrarán los aspectos más importantes de la funcionalidad del sistema atendiendo a la división de paquetes utilizada en la elicitación de requisitos: gestión de usuarios, gestión de bebidas, gestión de recetas y gestión del robot.

Para obtener más información sobre este apartado, consultar el anexo "*Anexo V. Manual de usuario".*

6.10.1 GESTIÓN DE USUARIOS

La aplicación dispone de una pantalla de inicio de sesión donde un usuario que no haya iniciado sesión dentro del sistema podrá hacerlo ("*Ilustración 54")*.

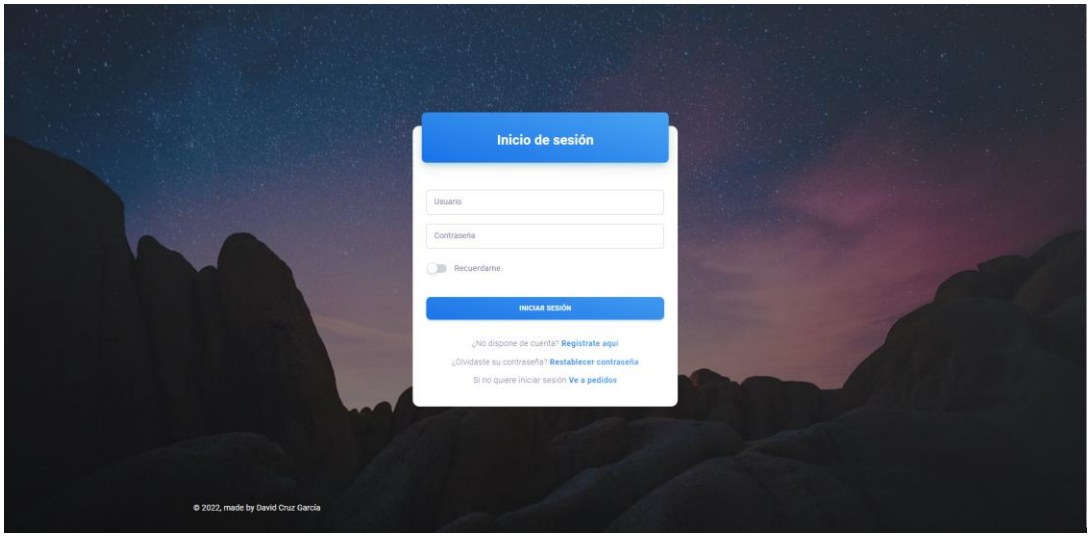


Ilustración 54. Pantalla de inicio de sesión.

En caso de no estar registrado en el sistema, se podrá registrar desde la pantalla de registro ("*Ilustración 55")*.

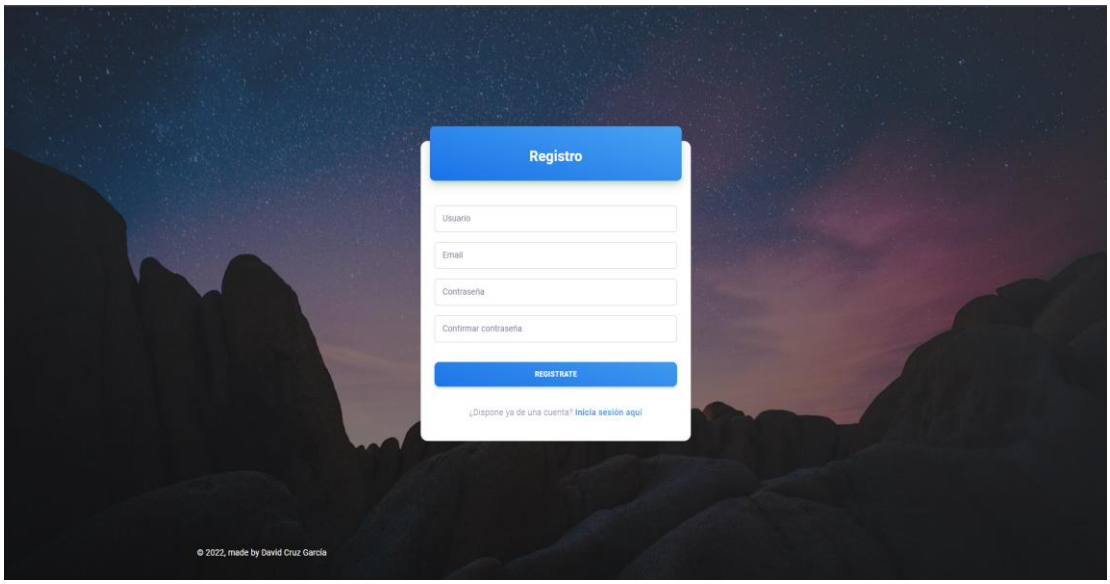


Ilustración 55. Pantalla de registro.

En el caso de que se olvide por alguna razón su contraseña, podrá recuperarla introduciendo su email en la pantalla de recuperación de contraseña (“*Ilustración 56”)*.

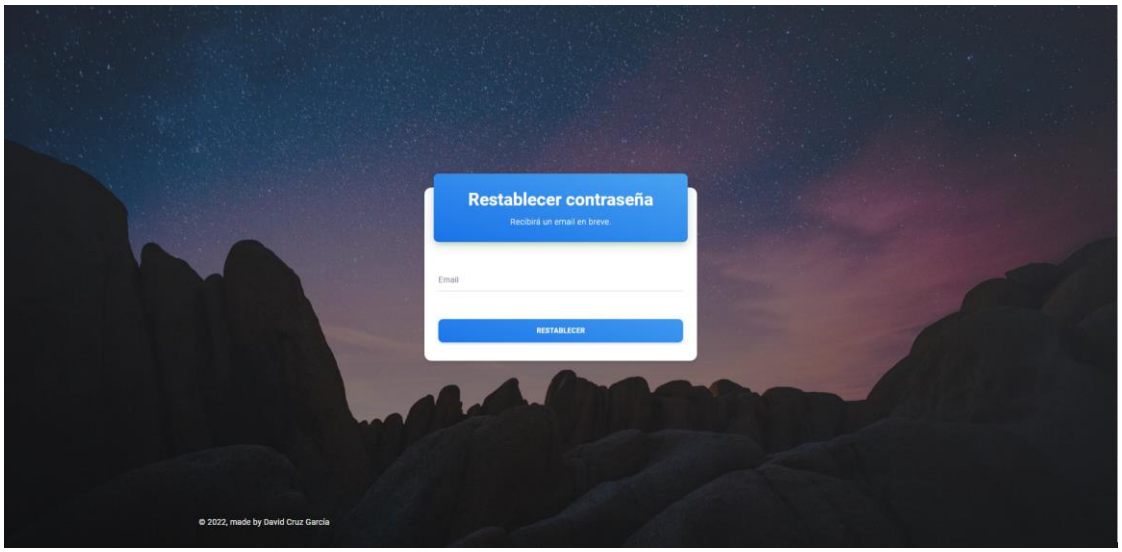


Ilustración 56. Pantalla restablecer contraseña.

Una vez el cliente haya iniciado sesión en el sistema (ver *"Ilustración 57"*), podrá realizar diferentes acciones. Centrándonos en la gestión de usuarios, podrá realizar 3 acciones:

Ilustración 57. Pantalla pedidos móvil tras inicio de sesión.

- ***Consultar su perfil***: accederá a su perfil para consultar su información y opciones (*"Ilustración 58"*).

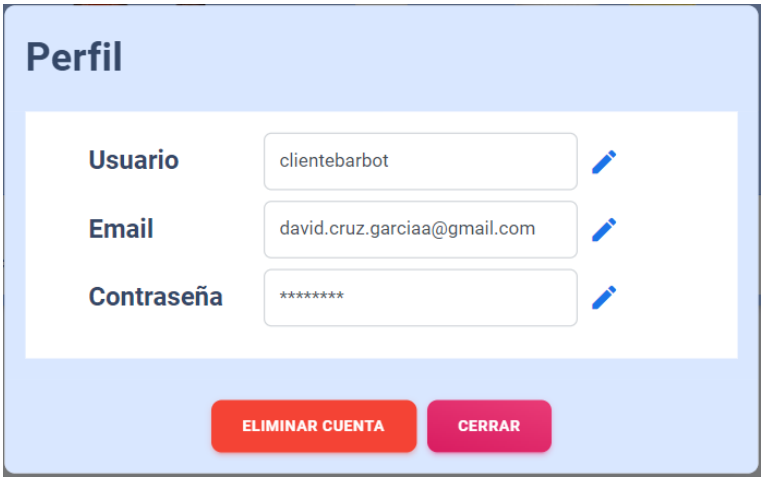


Ilustración 58. Perfil cliente.

- ***Consultar estadísticas***: el usuario podrá consultar sus estadísticas personales (“*Ilustración 59”).*

Ilustración 59. Estadísticas del cliente.

- ***Cerrar sesión***: cerrará sesión y accederá como un usuario no logueado.

En cuanto al administrador, podrá realizar las mismas opciones que el cliente, pero dispondrá además de un panel de control para la gestión de la aplicación y el robot (ver *“Ilustración 60” y “Ilustración 61”*).

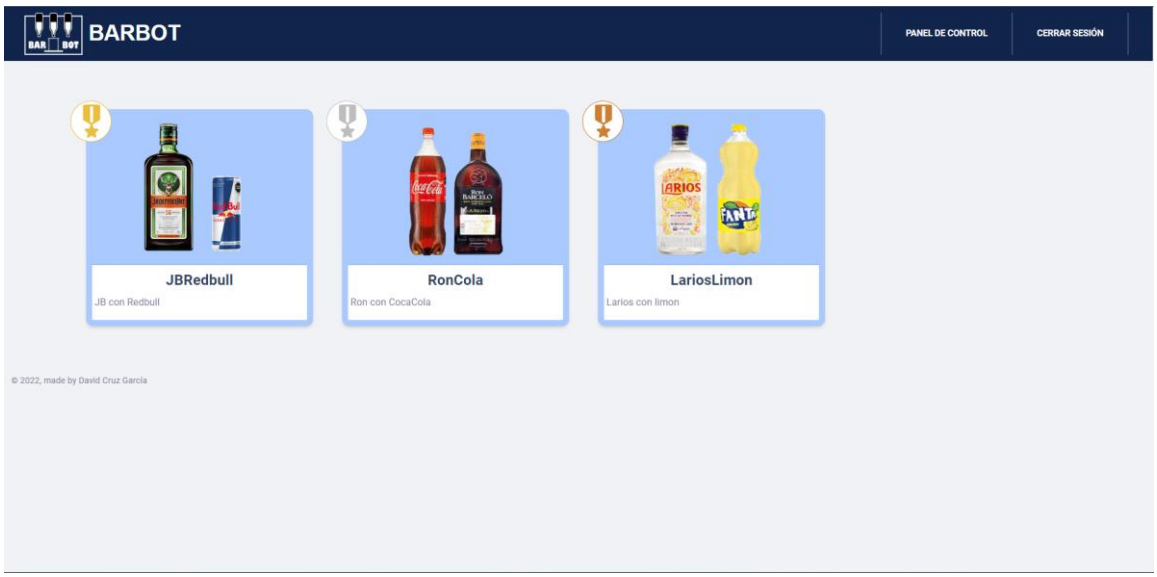


Ilustración 60. Pantalla administrador.

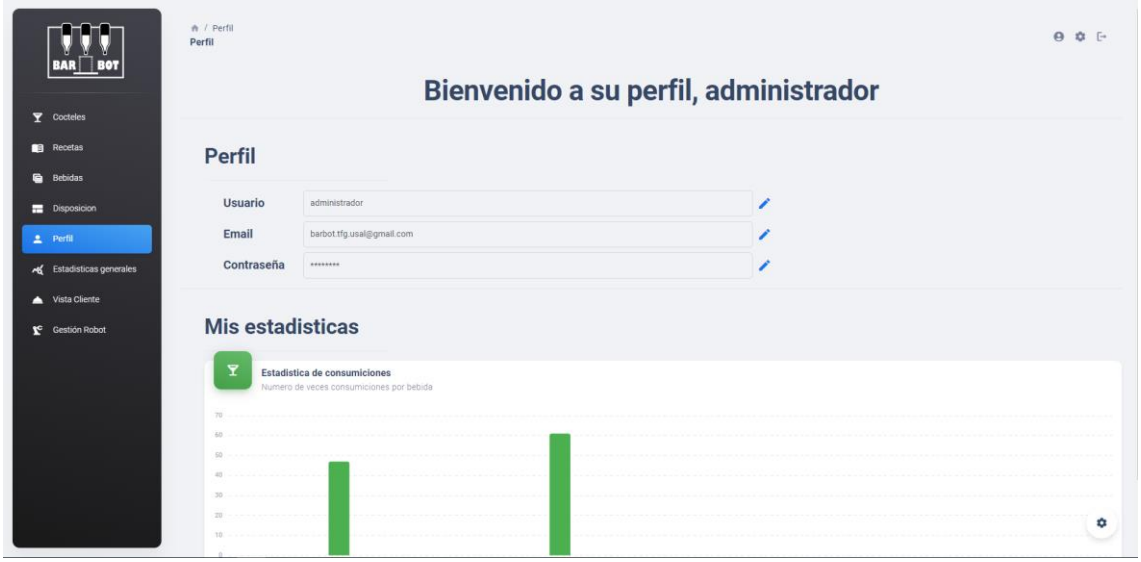


Ilustración 61. Perfil administrador.

6.10.1 GESTIÓN DE BEBIDAS

Una vez el usuario administrador ha iniciado sesión, podrá realizar varias acciones relacionadas con las bebidas:

- ***Añadir/Eliminar/Modificar bebidas:*** El administrador dispondrá de una pantalla con la lista de bebidas para poder gestionarlas (ver *"Ilustración 62"*).

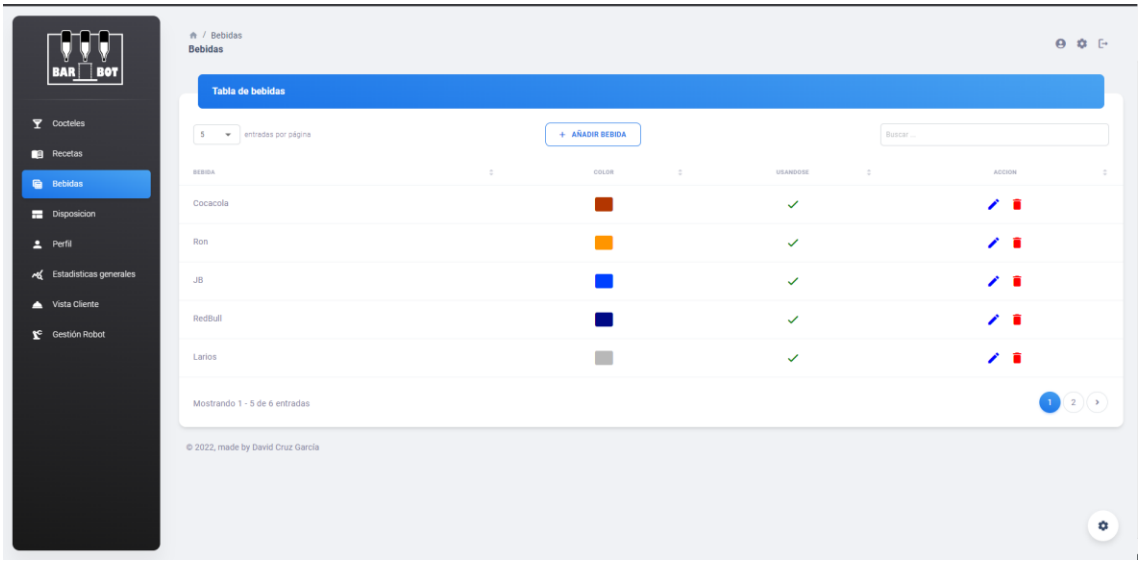


Ilustración 62. Pantalla de bebidas.

- ***Editar la disposición de bebidas en el robot:*** podrá indicar qué bebidas están colocadas en la disposición del robot (ver "*Ilustración 63").*

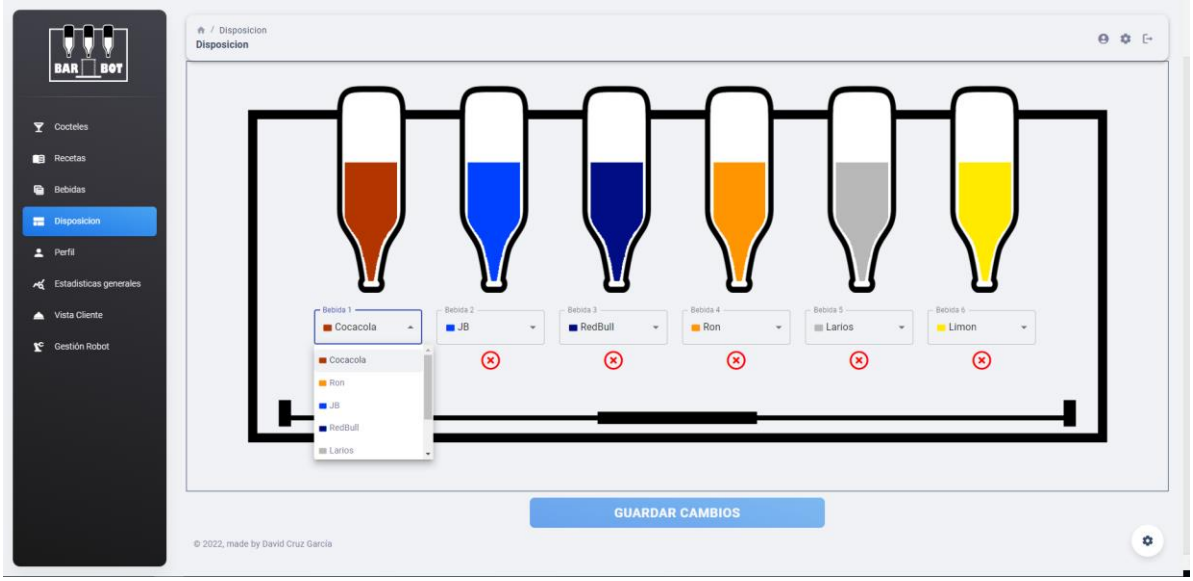


Ilustración 63. Pantalla disposición para ordenado.

6.10.2 GESTIÓN DE RECETAS

Una vez el usuario administrador ha iniciado sesión, podrá realizar varias acciones relacionadas con las recetas:

- ***Añadir/Eliminar/Modificar recetas:*** El administrador dispondrá de una pantalla con la lista de recetas para poder gestionarlas (ver *"Ilustración 64"*).

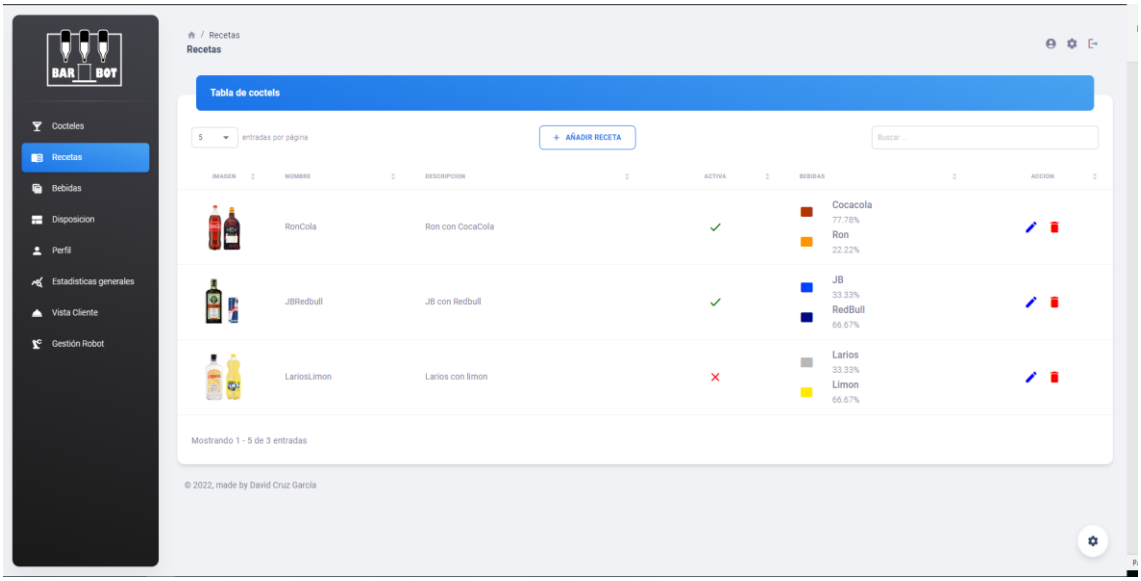


Ilustración 64. Pantalla recetas.

- ***Asignar porcentajes de bebidas a la receta:*** el administrador podrá asignar y modificar los diferentes porcentajes de las bebidas en las recetas (ver *"Ilustración 65"*).

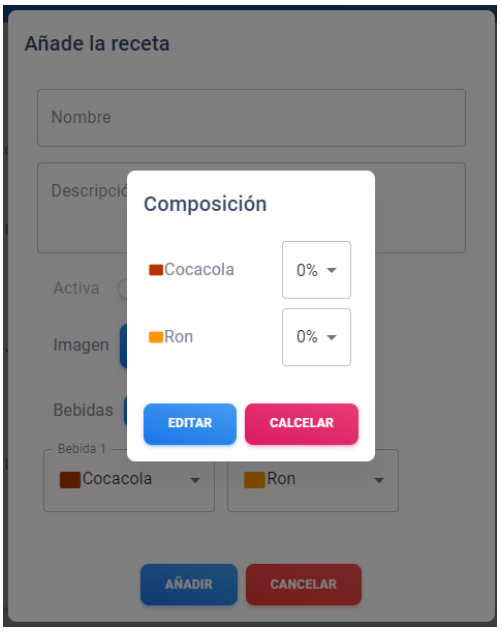


Ilustración 65. Composición receta.

Todos los usuarios podrán solicitar la preparación de una bebidas desde la pantalla principal de pedidos, además de poder consultar las bebidas que componen una receta (*ver "Ilustración 66"*):

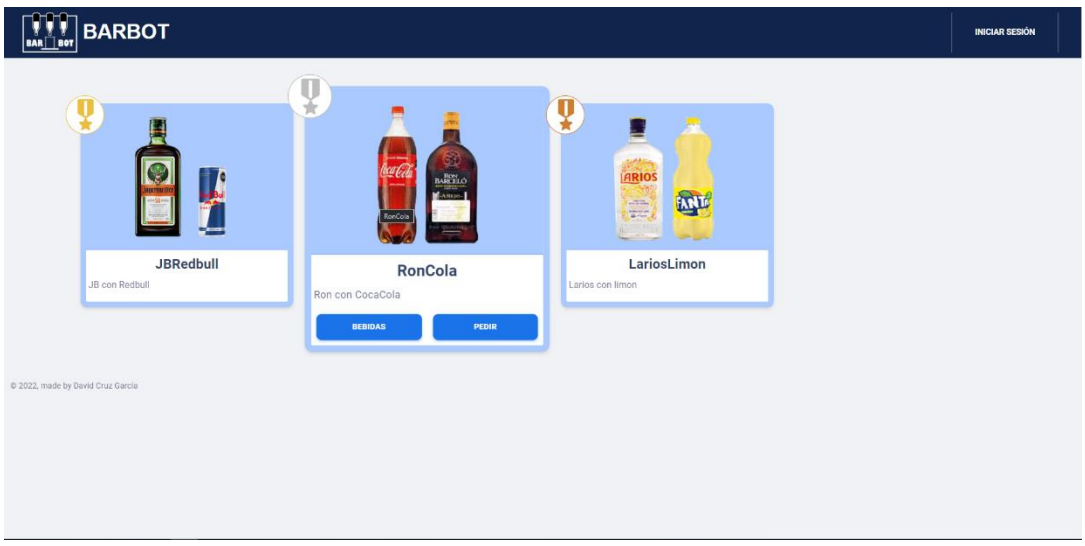


Ilustración 66. Pantalla pedir receta y ver bebidas.

6.10.3 GESTIÓN DEL ROBOT

Para la gestión del robot, el administrador podrá acceder al panel de control de Octoprint (ver "*Ilustración 67"*):

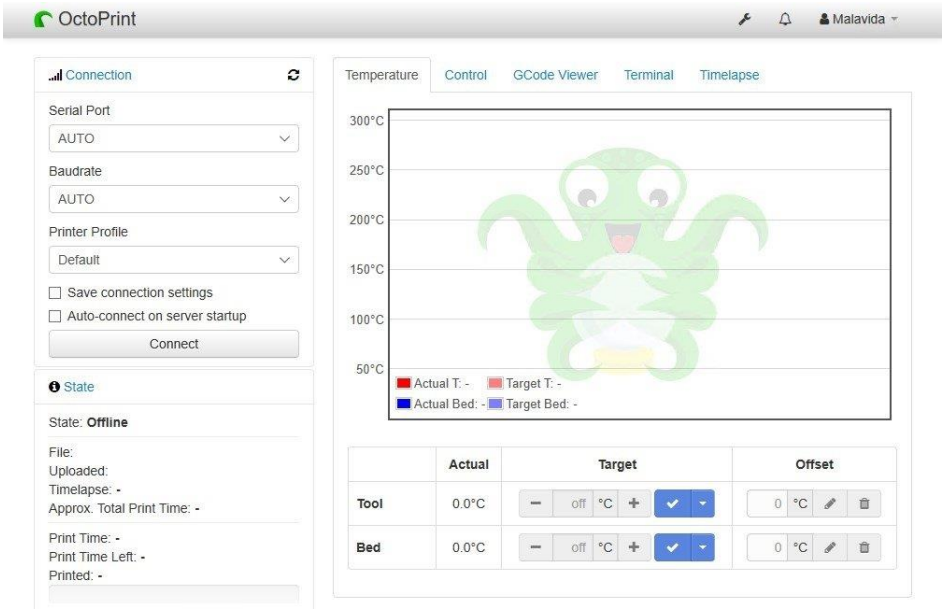


Ilustración 67. Panel de control del robot.

Además, el robot dispone de un led de aviso para indicar cuando se está preparando una bebida (rojo) o cuando está lista (verde), mostrados en la figura "*Ilustración 68"*.

Ilustración 68. LEDs preparación/finalización receta.

Para poder cambiar las botellas en el robot, basta con seguir los siguientes pasos (explicados con detenimiento en el manual de usuario):

7 LIMITACIONES DEL SISTEMA

Durante el desarrollo de este proyecto se han encontrado una serie de limitaciones que posiblemente pueden ser corregidas con una actualización en el diseño tanto del robot como de la aplicación o incluso están corregidas, pero no de la manera óptima:

- La longitud del robot y la dureza de los dispensadores al ser presionados provocan que el robot flexe cuando se encuentra en la parte central de las barras que actúan como guía del robot. La solución llevada a cabo es la fabricación de dos piezas de madera colocadas a milímetros bajo el robot a lo largo de toda la barra. Con esto se consigue que una vez el robot haga fuerza a la hora de dispensar una bebida se apoye en esta barra y pueda accionar el dispensador correctamente. Una mejor solución sería diseñar un robot que se mueva muy cerca de la base y utilizar unas barras de acero más resistentes.

- Pese a que se dispone de un firmware con una gran cantidad de opciones no se dispone de ningún método de feedback en el caso de avisar cuando una serie de comandos han sido ejecutados por el robot. Esto provoca que no se pueda implementar un mecanismo software capaz de bloquear el acceso a los pedidos cuando el robot ya está preparando un coctel. Hay 2 posibles soluciones:
    - Buscar otro firmware compatible con Arduino y Octoprint que ofrezca más opciones de feedback, o incluso diseñar uno personal, lo cual requiere conocimientos en electrónica para el uso correcto de los motores paso a paso.
    - Utilizar un componente hardware, como otro final de carrera que permita determinar cuándo una receta ha finalizado su proceso de preparación.

- Ya que el robot se controla mediante un servidor web, este requiere de acceso a internet para la comunicación con los servidores, por lo que puede limitar su uso en algunos espacios. La solución a esto sería plantear el desarrollo de otra aplicación de móvil o escritorio que comunique con el robot por bluetooth para el caso en el que no se disponga de internet.

8 LÍNEAS DE TRABAJO FUTURAS Y CONCLUSIONES

En este apartado se recogerán las conclusiones obtenidas tras finalizar el desarrollo de todo el proyecto y se plantearán posibles líneas de trabajo futuras para mejorar el diseño del sistema.

8.1 CONCLUSIONES

Una vez finalizado el proyecto se puede asegurar que se han cumplido con los objetivos establecidos al inicio del desarrollo del proyecto, así como se han satisfecho todos los requisitos planteados.

Los aspectos más destacables de este proyecto son:

- Aprendizaje y desarrollo de 2 servidores utilizando tecnología web no vista en el grado siguiendo una arquitectura Frontend-Backend. Este proyecto me ha permitido mejorar enormemente mis conocimientos sobre el desarrollo web.
- Diseño y construcción de un robot conectado a la aplicación, permitiéndome mejorar mis conocimientos de electrónica, modelado 3D y construcción. La electrónica es un hobbie que tengo, por lo que poder realizar mi trabajo de fin de grado mezclando la electrónica como la programación ha sido algo único.
- Puesta en práctica de los conocimientos adquiridos en las diferentes asignaturas del grado para el diseño de interfaces, diseño de bases de datos, gestión de proyectos software, etc.
- Se ha desarrollado un robot capaz de preparar bebidas y recetas de forma autónoma, permitiendo una total gestión del brazo robótico implementado.
- La aplicación permite el alta, la baja y la modificación de datos de usuarios, recetas y bebidas de forma que todas ellas están conectadas.
- El diseño del robot permite una personalización en el tipo de bebidas a utilizar y la disposición de estas en los distintos dispensadores, indicándolo en la propia aplicación.
- La aplicación y el robot permiten el almacenamiento y visualización de estadísticas, tanto personales de cada usuario como globales calculadas a partir de las personales, de forma anónima.

8.2 LÍNEAS DE TRABAJO FUTURAS

Ya que ningún diseño es perfecto, se plantean una serie de ideas para mejorar el sistema completo:

- Implementación de un sistema de dispensado automático de vasos de plástico (o papel teniendo en cuenta los problemas mundiales surgidos por el uso del plástico) que permita a los usuarios no tener ni que preocuparse por la colocación del vaso, además de la necesidad de disponer de uno.

- Implementación de un sistema de dispensado de hielos. Pese a que esta mejora se planteó, requería de un gran presupuesto y espacio para colocar una nevera fabricadora de hielo a partir de agua, además de que surge la necesidad de conectar dicha nevera a un suministro constante de agua.

- Uso de dispensadores de flujo libre o bombas peristálticas en lugar de dispensadores con depósito para mejorar la precisión en el dispensado de las diferentes bebidas. Con esta mejora se aceleraría el tiempo de dispensado de una bebida y la cantidad servida sería mucho más precisa.

- Mejoras en la aplicación, permitiendo a los clientes gestionar recetas personalizadas según las bebidas disponibles en el robot. Con esto, los clientes podrían dejar registradas sus recetas favoritas y, en caso de que las bebidas de dicha receta estén colocadas en el robot, podrían servirse los cocteles a su gusto.

Finalmente, estas líneas futuras pueden ser ampliadas realizando un pequeño estudio en las necesidades de los usuarios centrándonos en el sector hostelero y la preparación de bebidas, así como la posibilidad de ofrecer una sección de sugerencias e ideas para mejorar este prototipo.

9 REFERENCIAS Y BIBLIOGRAFÍA

- Rumbaugh, J., Jacobson, I., Booch, G. “El Lenguaje Unificado de Modelado. Manual de Referencia”. 2ª ed., Addison-Wesley. 2007
- Pressman, R. S. “Ingeniería del Software: Un Enfoque Práctico”. 7ª Edición. McGraw-Hill. 2010.
- Tuya, J., Ramos, I. y Dolado, J. (eds.) Técnicas Cuantitativas para la Gestión de Proyectos, Netbiblo, 2007.
- Documentación de Microsoft Project (visitado el 18-05-2022)
  - https://docs.microsoft.com/es-es/project/
- Jacobson, I., Booch, G., Rumbaugh, J. “El Proceso Unificado de Desarrollo de Software”. Addison-Wesley, 2000
- Documentación DrawIO (visitado el 20-05-2022)
  - https://drawio-app.com/tutorials/
- Documentación sobre patrones (visitados el 31-05-2022)
  - https://rlbisbe.net/2019/08/07/el-patron-bff-backend-for-frontend/
  - https://profile.es/blog/patrones-de-diseno-de-software/#Builder_Patterns
  - https://www.mongodb.com/docs/atlas/app-services/authentication/custom-jwt/
- Documentación SketchUp (visitado el 20-02-2022)
  - https://www.sketchup.com/taxonomy/term/296
- Documentación Arduino (visitado el 11-03-2022)
  - https://www.arduino.cc/reference/es/
- Documentación Ramps 1.4 (visitado el 11-03-2022)
  - https://reprap.org/wiki/RAMPS_1.4/es
- Documentación Raspberry PI (visitado el 5-03-2022)
  - https://www.raspberrypi.com/documentation/