UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación web para la gestión y reducción del desperdicio alimentario en la hostelería

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Soriano López, José Luis

Tutor/a: Valderas Aranda, Pedro José

CURSO ACADÉMICO: 2023/2024

Resum

En el sector de l'hostaleria es consumeixen milions de quilos o litres d'aliments i begudes a l'any, i tot el que no arriba a consumir-se es malgasta a causa d'una gestió deficient dels recursos per part de l'empresa hostelera i a un descontrol que es fa patent en hores punta del servei. En aquest treball es presenta una solució en forma d'aplicació web que permetrà a l'administrador donar d'alta els productes que posarà a la venda llestos per a consumir, com entrepans, plats cuinats o pastisseria, amb el seu preu, la quantitat d'ingredients utilitzats (que també donarà d'alta un per un), etc. Una vegada estiga tota la carta pujada, es podran registrar les comandes del client des d'un TPV. Amb cada comanda confirmada, l'aplicació guardarà un conjunt d'informació en la base de dades interna de l'empresa; dades amb les quals es prepararan gràfics i informes estadístics per a saber quins productes es consumeixen més i quins ingredients es gasten més que altres. L'aplicació permetrà la personalització segons les necessitats de cada empresa, que podrà pujar el seu logo, la seva tarifa i fotos de tots els seus productes si així ho considera necessari.

Paraules clau: Clima, React, ReactJS, Redux, Cypress, Hostaleria, Medi ambient, Agile, Desenvolupament web

Resumen

En el sector de la hostelería se consumen millones de kilos o litros de alimentos y bebidas al año, y todos los que no llegan a consumirse se desperdician debido a una gestión deficiente de los recursos por parte de la empresa hostelera y a un descontrol que se hace patente en horas punta del servicio. En este trabajo se presenta una solución en forma de aplicación web que permitirá al administrador dar de alta los productos que pondrá a la venta listos para consumir, como bocadillos, platos cocinados o bollería, con su precio, la cantidad de ingredientes utilizados (que también dará de alta uno por uno), etc. Una vez esté toda la carta subida, se podrán registrar los pedidos del cliente desde un TPV. Con cada pedido confirmado, la aplicación guardará un conjunto de información en la base de datos interna de la empresa; datos con los que se irán preparando gráficas e informes estadísticos para saber qué productos se consumen más y qué ingredientes se gastan más que otros. La aplicación permitirá la personalización según las necesidades de cada empresa, que podrá subir su logo, su tarifa y fotos de todos sus productos si así lo ve necesario.

Palabras clave: Clima, React, ReactJS, Redux, Cypress, Hostelería, Medio ambiente, Agile, Desarrollo web

Abstract

In the hospitality sector, millions of kilograms or liters of food and beverages are consumed annually, and all that is not consumed is wasted due to poor resource management by the hospitality company and a lack of control that becomes evident during peak service hours. This paper presents a solution in the form of a web application that will allow the administrator to register the products that will be ready for sale, such as sandwiches, cooked dishes, or pastries, along with their price, the quantity of ingredients used (which will also be registered one by one), etc. Once the entire menu is uploaded, customer orders can be recorded from a POS system. With each confirmed order, the application will save a set of information in the company's internal database; data that will be used to prepare graphs and statistical reports to know which products are consumed more and which ingredients are used more than others. The application will allow customization according to the needs of each company, which can upload its logo, pricing, and photos of all its products if deemed necessary.

Key words: Climate; React; ReactJS; Redux; Cypress; Hospitality; Environment; Agile; Web development

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

El desperdicio alimentario en la hostelería es un problema creciente que podría atajarse de raíz mediante una buena organización en las operaciones diarias de los negocios y con un uso eficiente de datos estadísticos. Así pues, con este proyecto se pretende estudiar y analizar cómo se podría alcanzar una solución útil y atractiva mediante una aplicación web. Para ello, se ha diseñado una app que facilita la buena gestión de los recursos humanos y materiales de la empresa. Además, guarda los datos estadísticos de los ingredientes y los productos más consumidos para construir tablas y gráficas para ayudar al establecimiento a llenar la cesta de la compra de un modo más eficiente. En la actualidad, hay restaurantes y bares tradicionales que, o no han considerado implementar un software que les ayude a potenciar sus operaciones, o si lo han hecho han tenido que abandonar la idea por la carga económica que supondría adaptarse a un modelo digital.

Según datos de la Oficina Europea de Estadística, en 2020 un 5 % de las toneladas de masa fresca que se desperdiciaron en España provinieron del sector de la hostelería, lo que supone un desperdicio aproximado de 4,5 kilos de comida por habitante. Puede parecer un porcentaje bajo, pero como se ve en la figura 1.1 la estimación fue de 213023 toneladas de alimento desperdiciado; un número demoledor. Y todo esto sin tener en cuenta el impacto que tuvo el COVID-19 en el año en que se realizó el estudio, ya que la reapertura de bares y restaurantes sugiere que esa cifra puede ser mucho mayor.

Por otro lado, en Europa, el desperdicio alimentario en 2021 en el sector hostelero fue de 12 kilos por habitante [Eur24], una cifra que está muy por debajo de los 70 kilos por habitante que se desperdicia en los hogares, o de los 28 kilos por habitante que se desperdicia en el sector de la manufacturación de alimentos también en Europa; pero, aun así, se trata de una cifra que debería y puede ser mucho menor.

Además, la FAO, la Organización de las Naciones Unidas para la Alimentación y la Agricultura, realizó un estudio en 2022 en el que determinó que el desperdicio alimentario ocasiona entre el 8 % y el 10 % de los gases de efecto invernadero [FAO24] . Medioambientalmente esto es insostenible, y es necesario que todo el mundo ponga de su parte para evitar un mal mayor. La repercusión que tiene todo esto no solo es medioambiental, sino que también tiene un impacto económico, con pérdidas estimadas en torno a los 132000 millones de euros a causa del desecho de alimentos, según los datos discutidos en un artículo que la Presidencia Española del Consejo de la UE publicó en 2023 [dlUE24].

	Total food waste	Primary production	Processing and manufacturing	Retail and other distribution of food	Restaurants and food services	Households
EU (')	*58 512 559*	*6 067 377*	*11 806 452*	*4 079 709*	*5 275 265*	*31 283 755*
Belgium	2 881 897	38 699	1 862 177	73 591	88 333	819 097
Bulgaria	596 844	228 472	156 435	15 708	14 375	181 854
Czechia	*972 445*	27 022	100 339	*64 394*	*37 941*	*742 749*
Denmark	1 286 488	66 452	596 599	99 500	62 544	461 392
Germany	10 922 321	190 203	1 612 505	762 352	1 860 980	6 496 282
Estonia	166 513	23 612	31 622	19 976	10 739	80 564
Ireland	770 316	70 413	219 453	60 894	178 507	241 048
Greece (')	2 048 189	*372 204*	375 158	*150 472*	*220 032*	930 323
Spain (')	4 260 845	*845 620*	1 419 257	*348 219*	*213 023*	*1 434 726*

Figures in italic are estimates

(') Definition differs in some figures

Source: Eurostat (online data code: env_wasfw)

eurostat

Figura 1.1: Desperdicio alimentario por actividad y sector en 2020

Existen muchos dueños de establecimientos dedicados a la restauración que no son conscientes de que, con un buen software, se pueden beneficiar tanto ellos como el mundo, ya que una buena gestión puede ayudar a reducir costos, mejorar la eficiencia operativa y contribuir a la sostenibilidad. Nos hallamos ante un problema titánico que debe afrontarse desde distintos ángulos para poder ser abordado totalmente, y aunque este trabajo se centra en buscar soluciones para el sector hostelero, esta no sigue siendo más que una de las muchas ramificaciones del problema. Para el resto de sectores existen varios trabajos muy interesantes, como por ejemplo el de Bárbara Sánchez, que propuso una metodología para que empresas como Mercavalencia siguieran ciertas estrategias que permitieran reducir el desperdicio alimentario en su Trabajo Fin de Master por la UPV: "*Reducir el Desperdicio Alimentario en Mercavalencia: indicadores para formular una estrategia*"[SL24].

Una búsqueda más avanzada de los trabajos dedicados al desperdicio alimentario en distintos ámbitos revela que no hay una carencia de soluciones, pero sí de personas que las vean atractivas y útiles para sus negocios; y el problema radica en encontrar una solución que alcance un compromiso que satisfaga a todas las partes. Es un deseo ferviente del autor de este trabajo que la aplicación web propuesta sirva de referente y de inspiración a futuras generaciones para desarrollar el problema desde otros punto de vista, o de proponer mejoras y soluciones más refinadas.

1.1 Motivación

Como ingeniero de software, pensar en cómo optimizar casi todas las facetas del día a día con una aplicación se ha convertido en una costumbre, y durante diversas visitas a varios locales de restauración se han observado momentos de puro caos: camareros corriendo de un lado para otro tomando comandas, platos equivocados que vuelven a cocina, y un descontrol que amarga la experiencia de trabajadores y comensales a la par. Por ello, en muchas de estas visitas siempre se plantea la misma cuestión: "*¿Cómo se podría mejorar el funcionamiento de este negocio con una buena aplicación informática?*".

En este proyecto se han aplicado todos los conocimientos adquiridos durante los estudios, especialmente los impartidos en la rama de Ingeniería del Software. Es importante tener en cuenta que la ingeniería informática es un campo muy amplio, y dispone de he-

rramientas potentísimas que pueden elevar al cuadrado la eficiencia de los negocios que estén dispuestos a apoyarse en ellas.

Una de estas herramientas es la disciplina conocida como Business Intelligence (BI), y que cada vez está cobrando más importancia entre las empresas. La inteligencia empresarial se basa en recoger todos los datos posibles de una organización: datos sobre los empleados, los gastos, los ingresos, los proveedores, etc.; en analizar cuidadosamente las tendencias, y finalmente en diseñar cuadros financieros con tasas y gráficas que ayuden a la empresa en la toma de decisiones.

El BI actúa como un mapa al futuro dibujado con datos del pasado, y las empresas más exitosas son las que implementan esta disciplina en sus negocios [Mic24]. Cuando se generan informes estadísticos mediante la Business Intelligence se le dota al propietario del restaurante de una abstracción del rendimiento de sus operaciones, y cuantos más datos recabe con el tiempo, más valor añadido aportarán dichos informes a su empresa. No es ninguna exageración decir que toda empresa que controle el pasado, dominará el futuro.

Con este trabajo se espera llamar la atención de negocios para que se considere la implementación del software propuesto como una oportunidad para crecer y contribuir en la defensa del medioambiente. Además, se pretende poner el foco de atención sobre una realidad: que sobra con pequeños recursos e inversiones para poder conseguir grandes logros.

1.2 Objetivos

El objetivo general este proyecto es desarrollar una aplicación web fácil sencilla y dinámica que ayude a cualquier profesional del sector de la hostelería a potenciar las operaciones de su negocio. Con esta aplicación y una PDA o un terminal se facilitará la toma decisiones a la hora de llenar la cesta de la compra mediante los datos estadísticos recogidos durante los servicios. Por otro lado, se pretende no solo mejorar los beneficios de la empresa, sino también contribuir a la sostenibilidad medioambiental combatiendo el desperdicio alimentario. Además, otro de sus objetivos es mejorar las condiciones laborales de camareros y del personal de cocina ofreciendo una ruta de operaciones óptima durante el desempeño de sus labores, lo que reducirá los errores y las equivocaciones que pueden contribuir también al desperdicio alimentario.

Para asegurar un desarrollo exitoso, se han fijado los siguientes objetivos específicos:

1. Diseñar la arquitectura de la aplicación web garantizando la escalabilidad y la eficiencia para negocios de todos los tamaños.
2. Desarrollar una interfaz de usuario intuitiva y personalizable que permita a los administradores dar de alta todos los productos de su negocio cómodamente.
3. Implementar una base de datos robusta para el almacenamiento y el tratamiento de toda información relacionada con las operaciones de un restaurante.
4. Integrar funcionalidades basadas en la Business Intelligence para el análisis de datos y la generación de informes y gráficos estadísticos que ayuden a identificar tanto las debilidades como las fortalezas de la empresa.

5. Realizar unas pruebas de validación heurística para verificar que se ha desarrollado una aplicación profesional de alta calidad.

1.3 Impacto Esperado

Así pues, se espera que la implementación de esta aplicación enfocada a reducir el desperdicio alimentario en la hostelería no solo suponga un ahorro considerable a largo plazo para todos aquellos restaurantes que hagan una inversión en ella, sino que también tenga un impacto positivo en la sociedad.

Gestionar eficientemente los recursos alimentarios resultará en una optimización de los costes y en el aumento de la rentabilidad de los negocios. Y no solo se verán beneficiados los propietarios, si no también el consumidor final, ya que este ahorro se puede traducir en menús con precios más competitivos. Por otro lado, el capital ahorrado se puede utilizar para invertir en otras áreas de mejora y expansión del negocio.

Además, la intención es que el proyecto contribuya a una mayor concienciación social respecto al consumo responsable y la sostenibilidad en la hostelería. Al implementar prácticas que reducen el desperdicio alimentario, los establecimientos gastronómicos pueden fomentar una cultura de responsabilidad y eficiencia entre sus empleados y clientes. Esto no solo mejorará la reputación de los negocios involucrados, sino que también promoverá hábitos sostenibles y responsables en la comunidad.

Así mismo, y no menos importante, la reducción del desperdicio alimentario contribuirá a la disminución de la huella ecológica que ejerce el sector. Los recursos bien gestionados son los que mejor minimizan el desperdicio, limitando así las emisiones de gases de efecto invernadero asociadas a la producción y eliminación de alimentos no consumidos. Este enfoque sostenible es crucial en la lucha contra el cambio climático y la preservación del medio ambiente.

Este proyecto está estrechamente ligado con los Objetivos de Desarrollo Sostenible de las Naciones Unidas [Uni24b], especialmente el "Objetivo 2: Poner fin al hambre", con el que podemos colaborar poniendo fin al desperdicio alimentario. Otro objetivo clave es el "Objetivo 12: Garantizar modalidades de consumo y producción sostenibles". Una de las metas de este objetivo, la 12.3, promueve que de aquí a 2030 hay que intentar '*reducir a la mitad el desperdicio de alimentos per capita mundial en la venta al por menor y a nivel de los consumidores*' [Uni24a].

Esta aplicación no solo pretende ofrecer una herramienta eficaz contra el desperdicio alimentario, sino que además espera conseguir que el día a día de camareros y cocineros sea más ameno, menos estresante y más enriquecedor. Es por ello por lo que también se adhiere al "Objetivo 8: Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos". Según datos de Randstat, que hizo un estudio de mercado en 2022, la tasa de rotación del personal de hostelería fue del 63,7 % [Ran24]. Se trata de un porcentaje preocupante, y que denota que hay negocios que pierden recursos humanos que podrían mantener con una buena gestión. Las condiciones laborales en un restaurante que opera al libre albedrío son duras, y si se guiara todo ese trabajo humano con una aplicación informática se podría mejorar no solo el rendimiento del negocio, sino también las condiciones de los trabajadores.

1.4 Estructura de la memoria

La memoria está estructurada en 10 capítulos y en varios anexos. Los cinco primeros capítulos de esta memoria cubren todo el proceso de investigación, diseño y organización del trabajo, y los últimos cinco cubren todo el proceso de desarrollo, implementación y pruebas. A continuación se ofrece un breve resumen de lo que se puede encontrar en cada capítulo:

Capítulo 1: Introducción Recoge una exposición de los motivos por los que se ha decidido realizar este proyecto y repasa la motivación que ha llevado a desarrollar la propuesta. Se detalla el objetivo general y los específicos, y el impacto esperado que este proyecto tendrá en la sociedad.

Capítulo 2: Estado del arte Un repaso por estudios y aplicaciones que abordan el mismo problema que este proyecto. Se indica en qué estado se encuentra hoy en día el tema de estudio y cuál es la propuesta de mejora que se plantea con en este trabajo.

Capítulo 3: Metodología Se explica la metodología seguida durante el diseño y las fases de desarrollo por las que ha pasado la aplicación en cada *sprint*, además de las distintas herramientas que se han adoptado de la filosofía *Ágile*.

Capítulo 4: Análisis de requisitos Detalles de cómo se han identificado los requisitos que soporta la aplicación y cuáles han sido los requisitos iniciales. Además, incluye los diagramas de casos de uso y los escenarios.

Capítulo 5: Análisis conceptual y diseño Se analiza el diagrama de clases, el modelo de la base de datos que utiliza el servidor y las pantallas prototipo de las interfaces de la aplicación.

Capítulo 6: Desarrollo de la solución Ahonda en profundidad en el proceso de desarrollo de la solución, la arquitectura en la que se basa el sistema, un breve contexto de las tecnologías utilizadas y ejemplos de código significativos.

Capítulo 7: Producto desarrollado Colección de capturas de pantalla del programa terminado y una explicación sobre lo que hace cada componente. Se detalla a fondo todos los paneles creados para los tres actores de la aplicación y lo que cada uno puede hacer en cada pantalla.

Capítulo 8: Validación Validación de la aplicación en el *backend* mediante pruebas automatizadas con Cypress y donde además se ponen a prueba las 10 reglas heurísticas de Jakob Nielsen.

Capítulo 9: Trabajo futuro Lista con propuestas de mejoras de las funcionalidades existentes y con otras funcionalidades que se quedaron en el *backlog*.

Capítulo 10: Conclusiones Resumen de la memoria y valoración personal del trabajo realizado.

Anexo A: Objetivos de desarrollo sostenible Tabla donde se indica el grado de relación del proyecto con los ODS y una reflexión sobre qué objetivos son más afines a la esencia del proyecto.

Anexo B: Glosario Lista de términos técnicos y sus definiciones para su consulta.

CAPÍTULO 2
Estado del arte

En este capítulo se identifican y se realizan varios análisis comparativos de los diferentes trabajos académicos que han abordado el problema del desperdicio alimentario en el sector hostelero. Es importante someter a análisis al estado del arte, ya que nos ayuda a entender el contexto actual del problema; y también someter las soluciones existentes a estudio, para asegurar que la solución desarrollada en este proyecto no se queda atrás.

Para ello, se han definido unos criterios de selección entre todos los trabajos académicos disponibles, y se ha seleccionado una de las tesis más interesantes que orbitan alrededor del dilema de cómo combatir la crisis del desperdicio de alimentos en el sector gastronómico mediante el uso de datos. Y, además, se echa un vistazo a cómo está el mercado de las aplicaciones que están diseñadas para combatir el desperdicio alimentario y sus estrategias para captar al usuario.

2.1 Revisión de trabajos académicos

El trabajo que vamos a analizar en esta sección es el de Don Diogo Xavier Ribeiro Pereira, que publicó una tesis de muy alto nivel proponiendo una solución para acabar con el desperdicio alimentario en la cafetería de su universidad empleando ingeniería de datos: "*Going zero waste in canteens: Exploring food demand with data analytics*", tesis que escribió para la Facultad de Ingeniería de la Universidad de Porto, en Portugal [Per24].

En su tesis, pone de manifiesto lo que seguramente sea el detalle más importante en el problema que se está planteando: que todos los propietarios de bares y restaurantes basan su proceso de aprovisionamiento según estimaciones propias y haciendo predicciones con dudosa eficacia. Esto lo que provoca es que el negocio se quede corto de ingredientes o que se quede con un excedente que, en el peor de los casos, se eche a perder. El estudio de Ribeiro se centró únicamente en una de las cafeterías de su universidad, y durante un período de dos años recogió todo tipo de variables, como por ejemplo los menús disponibles cada día, las condiciones meteorológicas o la cercanía de los períodos vacacionales; después, tras someter todos los datos a varios procesos, presentó toda la información sintetizada en gráficas para demostrar la validez de su estudio. Para ello, Ribeiro empleó distintas técnicas avanzadas de minería de datos como los Bosques Aleatorios (RFs), la Regresión con Vectores de Soporte (SVRs) o las Redes Neuronales Artificiales (ANNs).

Los RFs son un tipo de modelo de ensamble que emplea varios árboles de decisión para mejorar la precisión y reducir el riesgo de sobreajuste y son muy importantes para

el machine learning. Esta técnica ha demostrado tener una precisión y superioridad muy altas en comparación a otras metodologías similares [FGE14].

SVRs es método muy potente para abordar problemas de regresión estadística, y su papel es encontrar una función que tenga una desviación máxima de los valores reales del conjunto de entrenamiento. Esta técnica ofrece la ventaja de tener una calibración rápida y de otorgarle flexibilidad a la función de pérdida [Har17].

Los ANNs son modelos inspirados en la estructura del cerebro humano y están compuestos por nodos neuronales organizados en capas. Su objetivo es aproximar funciones complejas y detectar patrones en los datos que estudian. Estas redes neuronales artificiales fueron desarrolladas como la generalización de modelos matemáticos de sistemas nerviosos biológicos [Hop88].

Ribeiro almacenó todos los datos recabados durante su estudio en archivos de Excel y los transformó a código de software mediante el lenguaje de programación Python haciendo uso de varias librerías. Después, entrenó un modelo con los datos, comprobó su rendimiento con otra partición y utilizó varias métricas para poder medir el error de predicción del modelo. Tras poner en marcha y entrenar 36 modelos predictivos se quedó con los tres más precisos para predecir los consumos de carne, pescado y verduras cada día.

En una entrevista con el dueño de la cafetería, descubrió que el método que usaban para determinar las cantidades de ingredientes durante su aprovisionamiento era comparar las ventas del mismo día del año anterior. En la figura 2.1 se aprecia cómo el sistema predictivo que desarrolló en su tesis (barras azules) tenía un desperdicio de comida notablemente inferior al sistema más rudimentario que estaba empleando el establecimiento (barras naranjas).

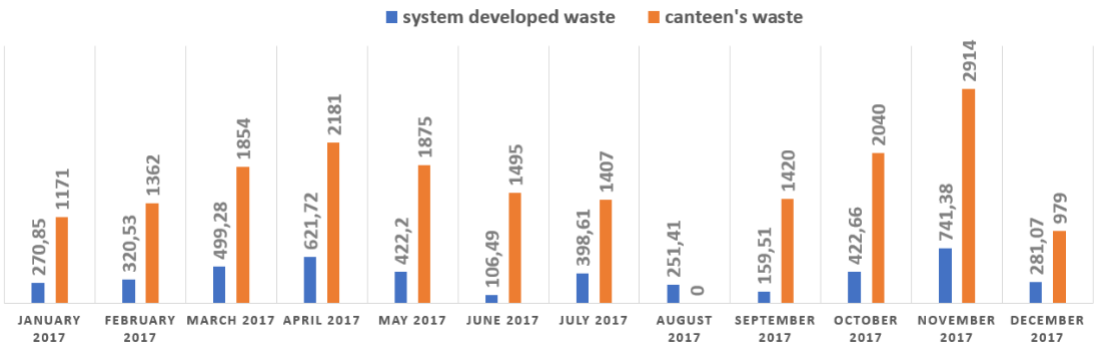

Figura 2.1: Comparación del desperdicio mensual generado por el sistema desarrollado en la tesis y por el método predictivo de la cafetería. De "*Going zero waste in canteens: Exploring food demand using data analytics*"(p. 40), por Diogo Xavier Ribeiro Pereira, Universidad de Porto, 2018.

Como dato curioso, el sistema no fue capaz de predecir que en agosto de 2017, a diferencia del año anterior, la cafetería iba a cerrar durante todo el mes; y propuso unas compras que habrían ocasionado pérdidas sustanciosas en el caso de que el sistema hubiera estado totalmente automatizado. Aun así, la tesis de Ribeiro puso de manifiesto que, con un buen estudio de los datos, se pueden conseguir auténticos logros en la reducción del desperdicio alimentario.

En comparación con el trabajo de Ribeiro, la aplicación desarrollada en este proyecto recorre otras soluciones. En primer lugar, pretende ser una herramienta cómoda para que los negocios registren en una base de datos absolutamente todas las ventas que hagan, y después con todos esos datos generar informes estadísticos para buscar tendencias. No se pretende dejar las decisiones de reabastecimiento a un algoritmo entrenado y privar al propietario de tomar decisiones propias, sino que se espera darle una herramienta con la que hacer estimaciones fundamentadas en datos y gráficas, sin menospreciar la experiencia que ha acumulado con los años en el negocio.

2.2 Revisión de apps comerciales

En lo que respecta a las aplicaciones informáticas creadas específicamente para combatir el desperdicio alimentario, actualmente en el mercado predominan las apps diseñadas para vender la comida que ha sobrado en un restaurante en el mismo día en que se han preparado, y poco antes de la hora de cierre. Entre ellas destacamos **To Good To Go** [Too24], una de las más reconocidas del mercado. Como se puede observar en la figura 2.2, la aplicación tiene una puntuación casi perfecta, de 4.8 estrellas, dispone de casi un millón y medio de reseñas y más de 50 millones de descargas.

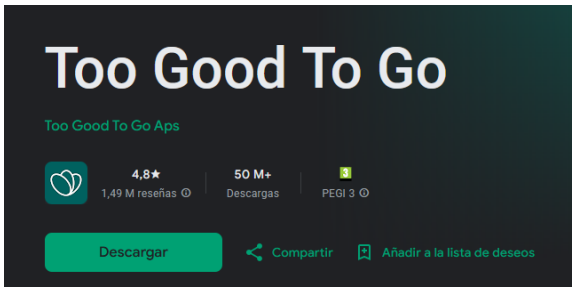

Figura 2.2: Estadísticas en Google Play Store de la aplicación Too Good To Go a fecha 22/06/2024.

Este tipo de aplicaciones tienen un planteamiento interesante y sus packs sorpresa son muy famosos, lo que demuestra que al ciudadano medio le preocupa el tema del desperdicio alimentario y quiere colaborar, a la par que ahorrar. No obstante, estas apps parten de un punto de salida totalmente distinto al enfoque y a la filosofía que sigue la aplicación de este proyecto. Esto se debe a que las apps como Too Good To Go atajan el problema de manera correctiva, y la app desarrollada en este trabajo lo ataja de manera preventiva. Ambas soluciones luchan contra lo mismo, pero cada en un orden diferente. A pesar de todo, estas aplicaciones y la estudiada en este trabajo pueden coexistir y beneficiarse mutuamente.

La viabilidad económica de todas las aplicaciones enfocadas a trabajar con restaurantes es muy alta. Según datos del INE, como se puede observar en la figura 2.3, en España en 2023 había registrados 263508 establecimientos gastronómicos, más de un cuarto de millón. La aplicación desarrollada en este proyecto tiene un alto potencial de rentabilidad debido a que existe un mercado amplio, una propuesta de valor clara, unos costes operativos bajos y una oportunidad de crecimiento y diversificación. Gracias a que pretende tener un impacto positivo en la sociedad y en el medio ambiente, la aplicación puede ser atractiva tanto para los consumidores como para los inversores.

En lo que respecta a las aplicaciones diseñadas para que los restaurantes creen su propia carta digital y ofrecer estadísticas de ventas a los clientes, tenemos a **Avocaty** [Avo24],

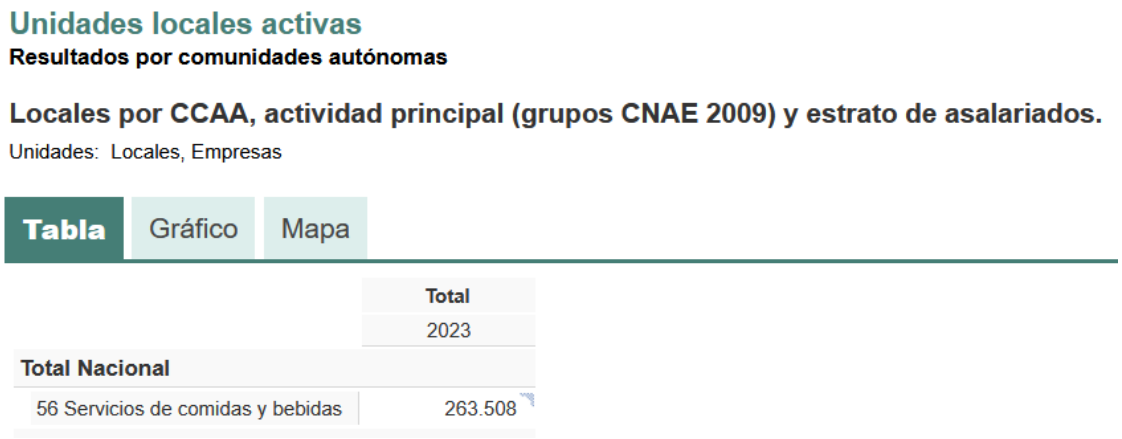

Figura 2.3: Número de establecimientos gastronómicos en España. Fuente: INE

una solución muy sólida para personalizar cualquier tipo de carta según las necesidades de cada negocio. Como se puede observar en la figura 2.4, se ha creado el negocio ficticio Shiroken Sushi en una cuenta de evaluación de la aplicación para ver cómo funciona.

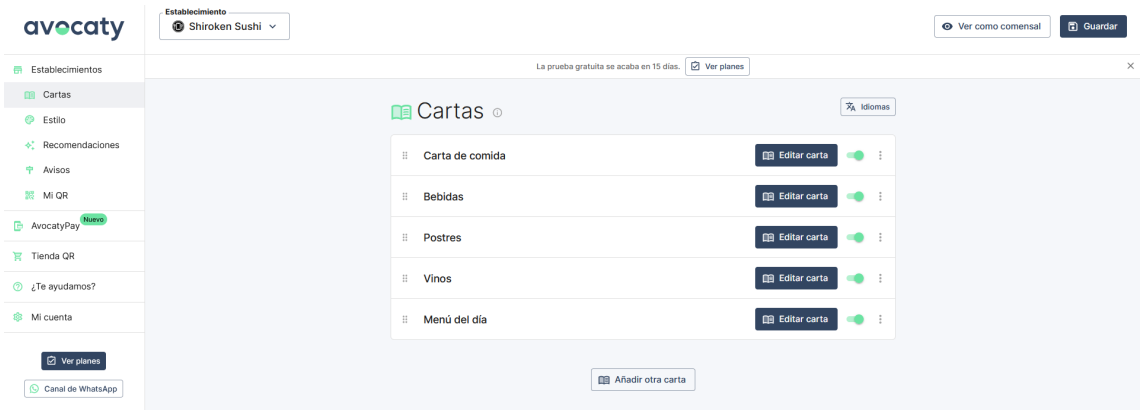

Figura 2.4: Interfaz de Avocaty para la creación de cartas digitales

Una vez creada la carta se accede a la sección de Comidas y se crea la sección Nigiris, donde se crea el ítem Salmón flambeado cuyo coste por unidad se ha fijado a 1.00€. El resultado se aprecia en la figura 2.5. La aplicación ofrece unas funcionalidades muy útiles para el propietario del negocio, y crear una carta es sencillo. Además, ofrece un servicio de recomendaciones basada en inteligencia artificial para que cuando el cliente esté preparando su pedido, el sistema le recomienden otros platos que haya podido pasar por alto.

La funcionalidad de Estadísticas de Avocaty sigue en desarrollo a fecha de hoy (Junio 2024), como indican en su web en la figura 2.6, por lo que no ha sido posible comprobar su funcionamiento. Avocaty es una aplicación web con un enfoque más comercial, y tiene planes de despliegue y mantenimiento según el tamaño y las necesidades de cada empresa. Además, no se especifica qué ocurre con el flujo de datos, a quién se informa de los pedidos, ni tampoco los roles de cada empleado.

En el caso de la aplicación desarrollada en este trabajo se ha propuesto una solución similar en cuanto a la creación de cartas, pero existirá un flujo constante de información de trabajo entre los empleados. En todo momento, los camareros encargados de servir las bebidas y los cafés sabrán cuándo un cliente ha pedido algo, y a qué mesa hay que

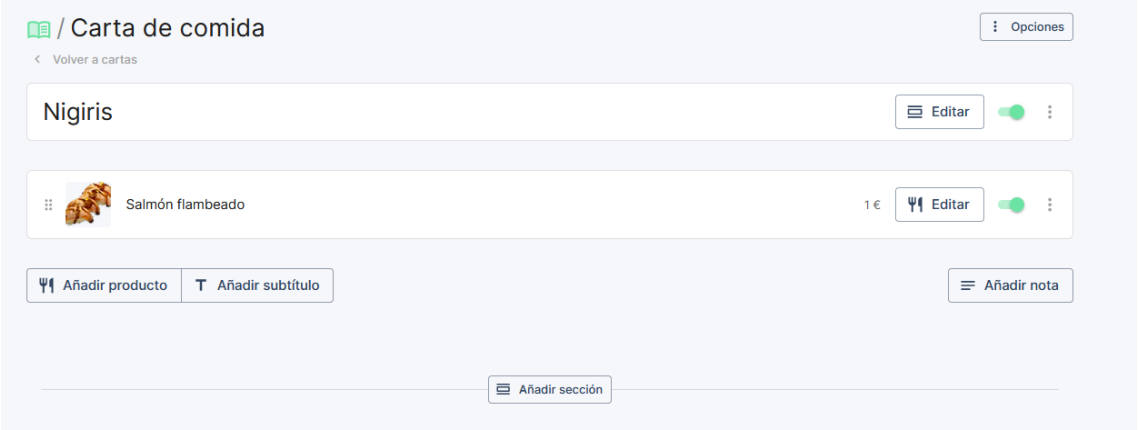

Figura 2.5: Interfaz de Avocaty donde se muestra la sección de Nigiris

Figura 2.6: La sección de Estadísticas de Avocaty aún en desarrollo

llevarlo. Cuando desde cocina marquen que el pedido de una mesa ya está listo para servir, le llegará la tarea al camarero encargado de servir los platos. Ya no es necesario que los camareros tengan acercarse a la cocina a entregar los nuevos pedidos, y a preguntar si ya han salido los anteriores. Es esa dinámica de trabajo la que se espera que convenza a los propietarios para implementar esta solución en sus negocios.

En resumen, la solución propuesta en este proyecto tiene varias funciones básicas que la diferencian de otras aplicaciones:

1. Optimizar al máximo las operaciones del restaurante y mantener una comunicación fluida entre los camareros y los cocineros.
2. Realizar un reparto de tareas organizado mediante roles de empleado y que evolucionará según el estado de los pedidos de cada mesa.
3. Almacenar los datos de todas las ventas y consumiciones en una base de datos.
4. Crear cuadros estadísticos mediante técnicas de Business Intelligence con los datos almacenados.

CAPÍTULO 3
Metodología

El uso de metodologías ágiles es de vital importancia para el desarrollo de software, y por ello se ha decidido utilizar el *framework Scrum* en este proyecto [Dep24], especialmente por su capacidad para gestionar proyectos complejos que se encuentran en constante evolución. Esta metodología prima por su enfoque iterativo e incremental, que permite la adaptación a cualquier tipo de cambio o contratiempo y asegurar así una entrega continua al cliente.

Se han definido una serie de *sprints* donde se ha repartido toda la carga de trabajo en tareas individuales definidas previamente en un *backlog*. Se partió desde la base de que habría elementos del *backlog* que no avanzarían a un *sprint* y se quedarían en espera hasta que se pudieran implementar en un futuro. Es irrazonable pensar que se puede llegar a cumplir todo lo que está incluido en el *backlog*, y esto se debe a distintas razones. Pero la más frecuente es la falta de tiempo, que nos obliga a aplazar algunas funcionalidades para otros *sprints* o para la siguiente iteración de la fase del desarrollo.

Durante la definición de cada *sprint* se ha sido realista con la carga de trabajo según el tiempo disponible, ya que se trata de un proyecto individual. En ambientes de trabajo profesional contamos con jefes de equipo que reparten las tareas, y con distintos especialistas: diseñadores, analistas de datos, programadores, etc. La carga se reparte a todo el equipo al inicio de los *sprints*, y los compañeros se ayudan entre ellos cuando hay alguna cuestión que les supera.

Por este motivo se ha decidido seguir una estrategia MVP, o mínimo producto viable, en el que se dejan apartadas la funcionalidades que no ofrecen valor a una demo funcional de un producto, como por ejemplo la pantalla de *login*, una sección de foro o chat, etc. Así pues, se han desarrollado los puntos fuertes de la aplicación, aquello por lo que destaca por encima del resto de aplicaciones estudiadas, y se ha dejado como trabajo futuro dotar a la aplicación de las funcionalidades más accesorias.

En resumen, en un proyecto de desarrollo de software individual debemos tener en cuenta dos factores clave, especialmente porque trabajamos sin ayuda de otros compañeros: las limitaciones técnicas y las limitaciones temporales. Aunque es cierto que el paso por el Grado de Ingeniería Informática en la UPV dota a los alumnos de una polivalencia muy alta, hay que saber elegir bien los desafíos y aprovechar al máximo las fortalezas de lo que se ha aprendido.

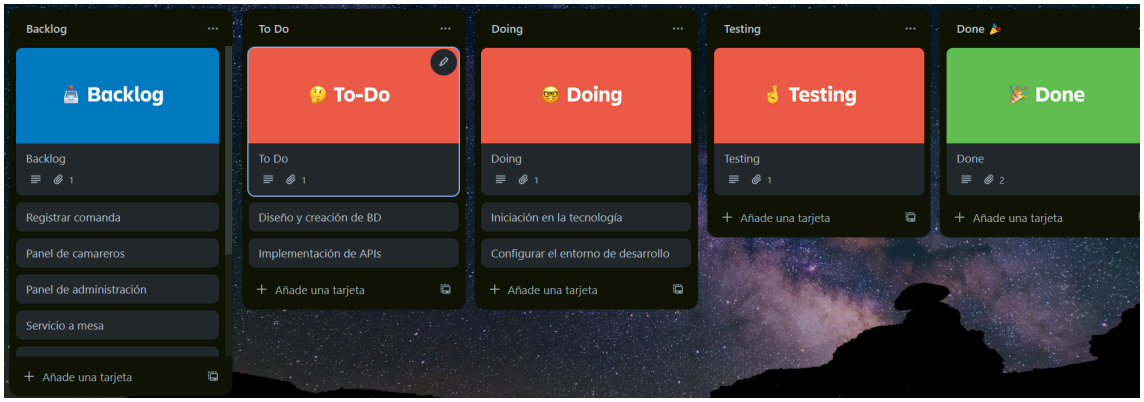

Figura 3.1: Tablero Kanban con las fases de desarrollo del proyecto.

Teniendo en cuenta las limitaciones expuestas, se ha seguido un proceso de desarrollo inteligente. Por ello, el proyecto está basado en una metodología SMART [Mar24], con objetivos Específicos (S), Medibles (M), Alcanzables (A), Realistas (R) y limitados por el Tiempo (T). A esto se lo conoce como *scrum* inteligente, y es una de las muchas claves del éxito de todo desarrollador de software.

Así pues, se comienza con una fase de planificación donde se define el alcance del proyecto, se identifican a los *stakeholders* o personas interesadas en el software a desarrollar y se elabora el *backlog* del producto mediante escenarios. Estas tareas deben ser redactadas con todo lujo de detalle, para que cuando se lean tanto hoy mismo como dentro de unas semanas su propósito siga estando claro. Para ello se ha construido el tablero Kanban que se aprecia en la figura 3.1 donde se puede ver el *backlog*, las tareas para hacer en ese sprint, las que se están haciendo, las que están en pruebas y las tareas finalizadas.

Después se organizan los períodos de los *sprints*, y dado el alcance del proyecto se decidió fijarlos en un período de una semana cada uno. Para aprovechar al máximo la metodología *Scrum*, a lo largo de cada semana se ha intentado hacer el esfuerzo de tener terminados todos los escenarios que se han programado. Dado que se ha contado con unas 20 horas semanales de dedicación al proyecto, se prepararon cinco *sprints* repartidos en dos meses, en los que se completaron las siguientes tareas:

1. **Sprint 1: Configuración Inicial y Diseño de la Base de Datos**: -Iniciación en las tecnologías mediante la lectura de libros. -Configuración del entorno de desarrollo. -Diseño y creación de un servidor de base de datos utilizando SQLite. -Diseño de una API básica para gestionar las acciones CRUD sobre la BD.

2. **Sprint 2: Desarrollo de la API y pantallas principales**: -Desarrollo e implementación de la API. -Validación y pruebas de la API con Cypress. -Creación de la estructura del proyecto con los componentes básicos para cada actor de la aplicación: paneles de camarero, de cocinero y de administrador.

3. **Sprint 3: Interfaces de usuario para cada componente de la aplicación**: -Diseño e implementación básica y sin funcionalidad de todas las secciones utilizando React. -Implementación de nuevas entradas en la API según las necesidades de la lógica de negocio de cada caso de uso. -Pantalla con el formulario para añadir y editar ingredientes finalizado. -Pantalla para listar las recetas y los productos de proveedores finalizado.

4. **Sprint 4: Gestión de comandas y actualizaciones en tiempo real**: -Desarrollo de la interfaz de usuario para registrar comandas. -Desarrollo de las interfaces para listar los pedidos listos para servir. -Desarrollo de la interfaz que muestra el estado actual de la mesas. -Implementación de WebSockets para actualizaciones en tiempo real en las pantallas de servicio de mesas y el terminal de cocina.

5. **Sprint 5: Implementación de estadísticas y refinamiento de las interfaces**: -Desarrollo de la API para obtener estadísticas básicas. -Implementación de gráficas y análisis en la interfaz de usuario. -Refinamiento de las interfaces mediante el uso de iconos y optimización de varias interfaces de usuario. -Pruebas finales y validación.

En la filosofía de trabajo de *Scrum* la planificación es vital, y las revisiones periódicas algo que no debe descuidarse. Cada comienzo de semana se ha realizado una *planning* donde se han decidido los siguientes pasos a seguir, y donde se ha llevado a cabo una revisión del trabajo que ya estaba terminado, una retrospectiva para determinar en qué aspectos se puede mejorar y una *daily*, o pequeña reunión diaria, de muy pocos minutos para ver dónde nos quedamos el día anterior, qué nos quedaba por delante y tomar medidas correctivas si habíamos tenido alguna dificultad con una tarea.

CAPÍTULO 4
Análisis de requisitos

El análisis de requisitos es un punto muy importante para evitar ir a ciegas durante el desarrollo de software, y hay que entender bien el problema que se va a abordar para encontrar una solución lo más efectiva posible. Todo este proceso se sigue para asegurar que el sistema cumpla con las expectativas del cliente y de los usuarios finales. A lo largo de este capítulo se detallarán los pasos seguidos para capturar los requisitos, la descripción de los requisitos iniciales y los casos de uso que ilustran las interacciones principales con el sistema.

4.1 Captura de requisitos

Para reunir información útil se contactó con el Restaurante la Gruta, ubicado en la localidad de Paiporta y famoso por recibir a comensales tan prestigiosos como los jugadores de fútbol del Valencia C.F. El negocio lleva abierto desde 1979 y está representado por una gerente con quien se mantuvo una reunión durante una visita al local. La reunión se celebró a principios de mayo, y tuvo una duración aproximada de 3 horas en las que se visitó el comedor donde están las mesas, la barra, la cocina y el almacén.

Durante la visita se le preguntó a la gerente cómo consideraba que podría mejorar el negocio con una buena aplicación informática, qué funcionalidades querría ver implementadas, y qué problemas tenía en ese mismo momento que esperaba resolver con el nuevo sistema. Además, se le preguntó en qué basaban su previsión de ingredientes a la hora de hacer la compra y cómo combatían el desperdicio alimentario.

De la conversación se extrajo que las compras las hace ya por costumbre, comprando siempre la misma cantidad de carne e ingredientes cada semana. Cuando hace falta más género llama a una empresa privada que se lo suministra, y los días que se han quedado sin producto no tienen más remedio que cerrar el local. En los días en los que sobra comida la distribuyen entre los trabajadores de la empresa, por lo que nunca se tira nada, pero la propietaria ha manifestado su interés por tener un control estadístico de las ventas y el consumo en su local para tomar decisiones de compra más acertadas.

Además de la entrevista con la máxima figura del negocio, la captura de requisitos se realizó también mediante un taller de trabajo con el equipo de cocina y mediante observación directa durante un servicio a la hora de comer en un fin de semana; uno de los días de más alta actividad para el restaurante. Una vez finalizado el servicio, se validó el siguiente caso de estudio con la gerente antes de dar por finalizada la visita.

4.1.1. Descripción general

El Restaurante la Gruta es un local de restauración de alto rendimiento donde se genera mucho movimiento en horas punta, y bajo esas condiciones la comunicación de los camareros con el personal de cocina y con quien prepara los cafés y las bebidas en la barra queda entorpecida a causa del ajetreo, lo que muchas veces ocasiona pérdidas de tiempo y retraso en los servicios. Los camareros van de aquí para allá, y muchas veces dependen de su memoria para saber qué mesa ha pedido qué cosa. El restaurante querría disponer de una aplicación que pueda usarse desde un ordenador o desde un terminal estilo tablet o PDA, y que parta de las siguientes premisas:

Los empleados dentro del restaurante tendrán roles, y puede haber rotación de roles cuando la gerente lo considere oportuno. El mismo rol puede asignarse a uno o a más camareros. La pantalla que verán en su PDA será distinta según el rol que tengan asignados.

El camarero que solo toma nota de las comandas cuenta con una tableta o PDA informática, y su papel principal es pasearse por las mesas anotando lo que van a beber, lo que van a almorzar o comer, y más tarde los cafés o si piden otro refresco u otra cerveza. Una vez ha confirmado la comanda, esta se transmite de inmediato al terminal del encargado de cocina, y al terminal del encargado del servicio a mesa. Cada mesa estará numerada, y cuando van a pagar la aplicación calculará la cuenta exacta de todos los consumos que se han registrado.

El camarero que se encarga del servicio a mesa puede consultar una lista de ítems pendientes de servir en su terminal y sabe a qué número de mesa tiene que llevarlo todo, y su papel además es recoger los botellines y las jarras vacías de las mesas. Estos pedidos son prioritarios por orden de llegada, es decir, el primero que ha pedido tiene prioridad. A los clientes les molesta ver cómo sirven a alguien que ha llegado más tarde que ellos. El encargado de este rol marca en el terminal que va a servir el pedido él, y cuando vuelva a revisarlo el sistema le mostrará en la lista las siguientes bebidas o platos que tiene que servir y a qué mesa van.

El jefe de cocina tiene un rol muy importante, ya que es el encargado de recibir los detalles de la comanda y de marcar en la PDA que ya tiene preparado un pedido listo para salir. Una vez marcado el pedido como preparado, se manda un aviso a la lista del servicio a mesa.

El propietario del local, o administrador, tendrá una pantalla donde se muestre el estado de las mesas, y sabrá cuál está libre, ocupada o reservada. Cuando le pidan la cuenta y seleccione una mesa podrá cobrarla y liberar la mesa, mandando la orden de limpiarla a los camareros responsables. Además, podrá dar de alta la carta digital del restaurante comenzando por las familias de artículos: Bebidas, entrantes, platos principales, postres, etc.

Cada familia tendrá su nombre y su descripción. Cada artículo tendrá detallado su nombre, su descripción, su precio y su tipo; y si la elaboración se realiza desde el mismo restaurante se especificarán los ingredientes necesarios para cocinarlo y las cantidades, si no, se indicará que el producto ya se compra manufacturado indicando quién es el proveedor o fabricante y el coste.

El administrador podrá ver el estado de las mesas. De cada mesa se almacena su número identificativo, la cantidad de comensales sentados, su estado (libre, ocupada o reservada), y su capacidad. También podrá dar de alta todos los ingredientes que necesita el restaurante en el día a día. De cada ingrediente se almacenará su nombre, descripción, cantidad en stock, unidad de medida, su coste de empresa, fecha de adquisición, fecha de caducidad, proveedor, categoría, alérgenos y observaciones. Cada producto del local puede estar elaborado con ningún ingrediente si es un producto ya manufacturado, o con muchos ingredientes si es algo que se tiene que preparar en la cocina.

La aplicación informática guardará datos estadísticos para saber qué ingredientes se gastan más, cuándo se bebe más cerveza, más vino, o más refrescos; y descubrir tendencias, saber en qué meses del año se consume algo más que en otro. Una vez se pueda determinar qué platos son más famosos y qué platos se piden menos, se podrá ajustar la carta. Lo que se pretende es saber qué ingredientes comprar, y hacer una previsión de las cantidades basándose en los datos reales que ha recogido el negocio, ahorrándose así un excedente que se podría echar a perder.

4.2 Requisitos iniciales

En esta sección se proporciona una descripción detallada de los requisitos que debe cumplir el sistema según el caso de estudio que se han validado con el cliente. Se divide en dos subsecciones según los tipos de requisitos: funcionales y no funcionales. Los requisitos funcionales describen las funciones específicas que debe realizar el sistema, y deben ser claros, concisos y cuantificables. Por otro lado, los requisitos no funcionales describen características de calidad que debe tener el sistema como, por ejemplo, el rendimiento, la seguridad, la usabilidad y la escalabilidad.

4.2.1. Requisitos funcionales

1. Gestión de Mesas
   - El sistema debe permitir registrar el número de comensales y la capacidad de cada mesa.
   - El sistema debe poder actualizar el estado de una mesa (libre u ocupada).
2. Gestión de Comandas
   - El camarero debe poder registrar una nueva comanda asociada a una mesa específica.
   - Cada comanda debe registrar la fecha y hora de creación de la comanda.
   - El sistema debe calcular el total de la cuenta de la comanda.
3. Gestión de Líneas de Comanda
   - El camarero debe poder añadir, modificar o eliminar líneas de comanda.
   - Cada línea de comanda debe registrar el artículo, la cantidad y el precio total.
4. Gestión de Artículos
   - El sistema debe permitir la creación, actualización y eliminación de artículos.
   - Cada artículo debe tener un nombre, descripción, precio y familia (bebida o plato).

- Los artículos pueden ser platos cocinados o productos manufacturados.

5. Gestión de Ingredientes
   - El sistema debe permitir la creación, actualización y eliminación de ingredientes.
   - Cada ingrediente debe tener un nombre, descripción, cantidad en stock, unidad de medida, coste, fecha de adquisición, fecha de caducidad, proveedor, categoría y alérgenos.
6. Gestión de Recetas de Platos Cocinados
   - El sistema debe permitir la creación, actualización y eliminación de recetas.
   - Cada receta debe vincular artículos con sus ingredientes y especificar las cantidades necesarias.
7. Consulta de Estadísticas
   - El sistema debe permitir al gerente consultar estadísticas de consumo de ingredientes.
   - El sistema debe permitir al gerente consultar la popularidad de los platos.
   - El sistema debe permitir al gerente consultar tendencias de consumo por mes.
8. Interacción del Camarero
   - El camarero debe poder tomar comandas y registrar los pedidos en el sistema mediante una tableta o PDA.
   - El camarero debe poder consultar y actualizar el estado de las mesas.
   - El camarero debe poder confirmar la entrega de los artículos pedidos.
9. Interacción del Gerente
   - El gerente debe poder gestionar la carta digital, incluyendo la creación, actualización y eliminación de artículos y familias de artículos.
   - El gerente debe poder gestionar los ingredientes, incluyendo la actualización del stock y la creación de nuevos ingredientes.
   - El gerente debe poder consultar y generar informes estadísticos para analizar el rendimiento del restaurante.

4.2.2. Requisitos no funcionales

1. Usabilidad
   - La interfaz debe ser intuitiva y fácil de usar para los distintos roles de empleados.
   - La aplicación debe ser accesible desde tabletas, PDAs y ordenadores.
2. Rendimiento
   - El sistema debe manejar eficientemente el flujo de datos durante las horas punta sin degradar el rendimiento.
   - Las actualizaciones de estado y pedidos deben ser en tiempo real.
3. Escalabilidad

- El sistema debe ser capaz de escalar para manejar un aumento en el número de mesas y pedidos.

4. Mantenimiento

   - El sistema debe ser fácil de actualizar y mantener, con soporte para nuevas funcionalidades y correcciones de errores.

5. Confiabilidad

   - El sistema debe ser fiable y estar disponible durante las horas de operación del restaurante.
   - Debe haber mecanismos para la recuperación de datos en caso de fallos del sistema.

6. Portabilidad

   - La aplicación debe poder ser desplegada en diferentes plataformas (Windows, iOS, Android) sin necesidad de cambios significativos.

7. Interoperabilidad

   - El sistema debe ser capaz de integrarse con otros sistemas de gestión de restaurantes y plataformas de pago.

4.3 Casos de uso y escenarios

En esta sección se detallan los casos de uso y los escenarios que ilustran cómo interactúan los diferentes roles definidos por el restaurante con el sistema propuesto. Los casos de uso describen las funciones específicas que el sistema debe realizar desde la perspectiva de los distintos usuarios [Uni22], y se trata de uno de los pasos más importantes durante el desarrollo de software, ya que ofrece un cuadro general de todas las funcionalidades de las que debe disponer la aplicación. Cada caso de uso va acompañado de un flujo de eventos que muestra los pasos necesarios para completar una tarea. Primero se presenta el diagrama de contexto, después se muestra el diagrama inicial y finalmente se detalla la solución completa con el diagrama estructurado [Uni18].

En lo que respecta a los escenarios, estos proporcionan ejemplos prácticos de cómo se llevarán a cabo estas tareas en situaciones reales en el restaurante. Estos escenarios ayudan a visualizar el funcionamiento del sistema en contextos específicos y demuestran cómo se optimiza la eficiencia y la comunicación entre los empleados.

4.3.1. Diagrama de contexto

El diagrama de contexto de la figura 4.1 nos ayuda a identificar los límites del sistema y a los actores. En nuestro caso de estudio se han identificado a tres actores esenciales, que son quienes interactuarán con la aplicación: la figura de camarero, el cocinero y el gerente. Es una imagen sencilla, en la que se representa el contexto de la aplicación y sirve para identificar de un vistazo el entorno en el que será utilizada.

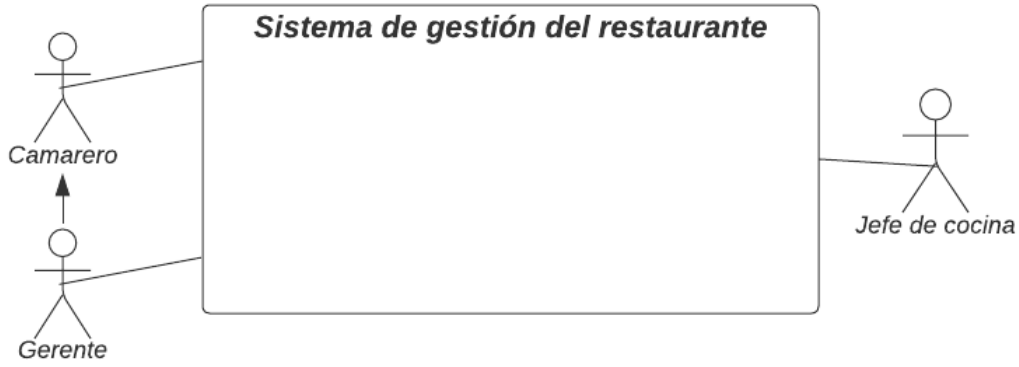

Figura 4.1: Diagrama de contexto de la aplicación desarrollada.

4.3.2. Diagrama inicial

El diagrama inicial de la figura 4.2 amplia la información del diagrama de contexto indicando todas las funcionalidades y qué actores interactúan con cada una de ellas. Nos da una idea más general de qué se puede hacer con la aplicación.

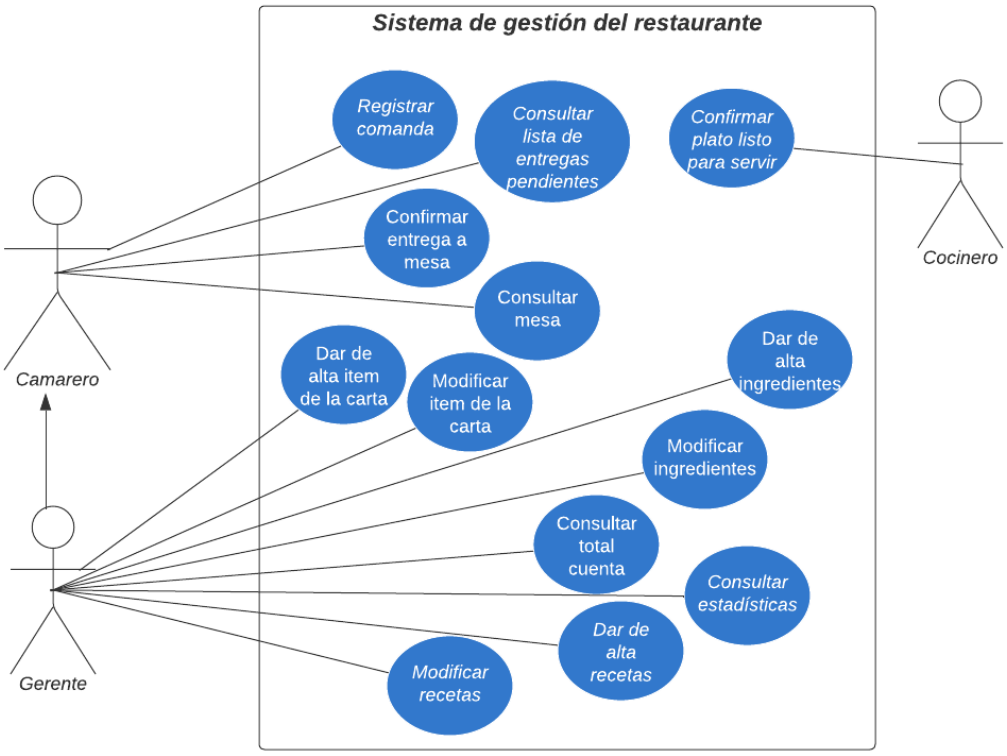

Figura 4.2: Diagrama inicial de la aplicación desarrollada

4.3.3. Diagrama estructurado

El diagrama estructurado o modelo de casos de uso que se puede ver en la figura 4.3 refina toda la información vista en los diagramas de contexto e inicial dando un vistazo al funcionamiento interno del sistema, y lleva los casos de uso a otro nivel de abstracción haciendo uso de relaciones de inclusión y de extensión. Los casos de uso incluidos significa que siempre se van a realizar junto con el caso de uso que los incluye, por otro lado, los casos de uso extendidos solo se realizarán si se cumple una condición.

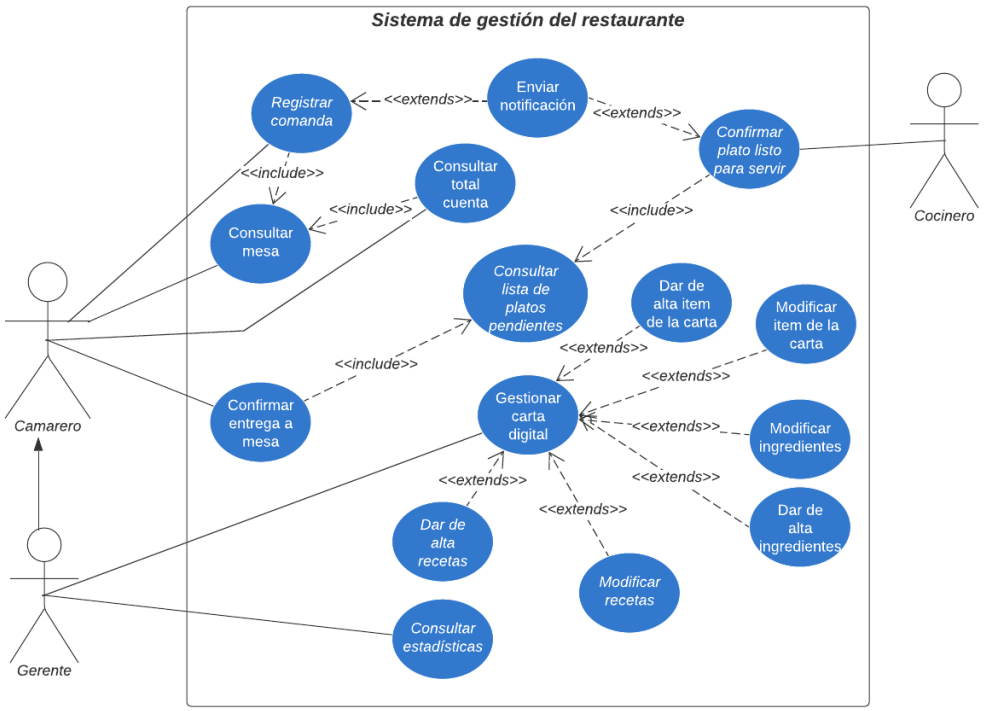

Figura 4.3: Diagrama estructurado de la aplicación desarrollada

En este diagrama se ven partes importantes del funcionamiento de la aplicación, como por ejemplo:

- Para registrar una comanda es necesario que el sistema consulte todas las mesas mostrando todos los artículos que acumulan estas.
- Para enviar una notificación el camarero tiene que confirmar una comanda o un cocinero debe confirmar un plato preparado.
- Para confirmar una entrega o un plato listo para servir, tanto los camareros como los cocineros necesitan consultar la lista de ítems pendientes.
- Que el gerente pueda dar de alta o modificar los distintos artículos de la carta dependerá de una acción general que es acceder a la gestión de la carta digital.

4.3.4. Escenarios

A continuación indicamos algunos de los escenarios más importantes de la aplicación:

- **Escenario 1: Registro de nuevas comandas en hora punta**
  - **Contexto**: Son las 14:00, el restaurante está lleno y el camarero Juan está tomando pedidos con su PDA.
  - **Flujo de eventos**:
    1. Juan selecciona la mesa 5 en su PDA.
    2. Los clientes en la mesa 5 piden 2 refrescos y dos platos.
    3. Juan registra el pedido en la PDA y lo confirma.
    4. La comanda se transmite automáticamente a la cocina y a la barra de bebidas.

- **Resultado**: La cocina y la barra de bebidas reciben la comanda sin retrasos, mejorando la eficiencia del servicio.

- **Escenario 2: Servicio de bebidas prioritario**
  - **Contexto**: Son las 15:00 y varios clientes han pedido bebidas y la camarera Ana las está sirviendo.
  - **Flujo de eventos**:
    1. Ana ve que hay un pedido de bebidas para la mesa 3 en su terminal.
    2. Ana confirma que las va a servir ella en el sistema, y las sirve en la mesa 3.
    3. El sistema muestra el siguiente pedido pendiente para la mesa 1.
  - **Resultado**: Las bebidas se sirven en orden de llegada, evitando molestias a los clientes.

- **Escenario 3: Preparación y servicio de platos**
  - **Contexto**: Son las 13:30 y la jefa de cocina, Paquita, está preparando los pedidos.
  - **Flujo de eventos**:
    1. Paquita recibe una comanda para la mesa 2.
    2. Paquita prepara los platos y marca en la terminal de cocina que están listos.
    3. La camarera Ana recibe una notificación en su PDA.
    4. Ana recoge los platos y los sirve en la mesa 2.
  - **Resultado**: Los platos se sirven puntualmente y los clientes quedan satisfechos con el servicio.

- **Escenario 4: Registrar elementos en la carta digital**
  - **Contexto**: El gerente abre el panel de administración y da de alta artículos en la carta.
  - **Flujo de eventos**:
    1. El administrador accede a la sección de gestión de la carta digital desde el panel de administración.
    2. Selecciona la opción Gestión de artículos para agregar una nueva familia de artículos.
    3. Introduce el nombre de los artículos, una descripción, precio y el resto de los campos de los que dispone información.
    4. Guarda el nuevo artículo.
    5. Si el artículo se elabora en el restaurante introduce los ingredientes necesarios y las cantidades.
    6. Repite el proceso para cada artículo adicional que desee agregar a la carta.
  - **Resultado**: Todos los productos que comercializa el restaurante están dados de alta en el sistema y se pueden agregar a las comandas.

CAPÍTULO 5
Análisis conceptual y diseño

En esta sección se realiza el análisis conceptual y el diseño del sistema, proporcionando una visión integral de su estructura y su funcionamiento. En este capítulo se incluye el diagrama de clases, que muestra la organización y las relaciones entre los distintos componentes del sistema; el modelo de base de datos que el servidor utilizará para almacenar y gestionar la información, y los bocetos de las interfaces de la aplicación, que sirven como guía para el desarrollo de la interfaz de usuario.

5.1 Diagrama de clases

El diagrama de clases es una representación gráfica de las clases del sistema y cómo están relacionadas entre ellas. Este diagrama ayuda a visualizar la estructura del sistema definiendo las clases, sus atributos y métodos, así como las relaciones de herencia, asociación y agregación. En la figura 5.1 se ve representado el diagrama de clases de la aplicación.

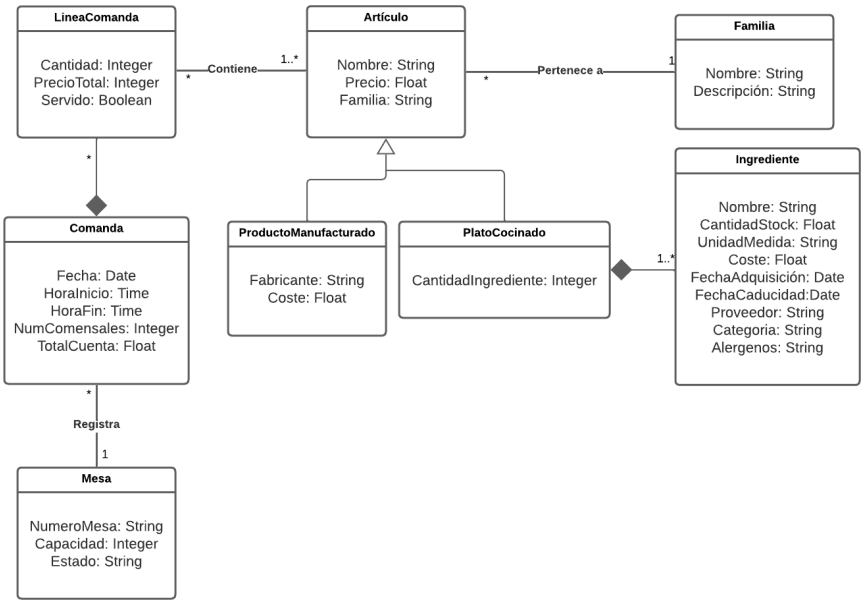

Figura 5.1: Diagrama de clases UML de la aplicación

5.2 Modelo de la base de datos

Un modelo de base de datos es una representación estructurada de cómo se organizan y se gestiona la información en un sistema gestor de bases de datos. Este modelo define las reglas y las estructuras que determinan cómo se almacenan, se relacionan y se manipulan los datos; y aunque existen muchos modelos de BD, el más utilizado es el modelo relacional.

Una base de datos relacional se organiza en tablas con claves primarias, claves ajenas y restricciones de integridad, lo que asegura la consistencia y la exactitud de los datos. Además, nos permite hacer consultas flexibles mediante el lenguaje de consulta estructurado SQL, y determinar el tipo de dato que se almacenará en cada atributo. A continuación se detalla el proceso de transformación del diagrama de clases del apartado anterior a tablas de bases de datos.

5.2.1. Descripción de las tablas

El sistema cuenta con 8 tablas para almacenar todos los datos necesarios para que la aplicación pueda funcionar adecuadamente, y son las siguientes:

5.2.2. Tabla Mesa

La tabla **Mesa** representa a cada una de las mesas del restaurante. Está compuesta por los siguientes atributos:

- **NumeroMesa**: identificador único de la mesa (String).
- **Capacidad**: la capacidad máxima de la mesa (Integer).
- **Estado**: el estado actual de la mesa (puede ser 'libre' u 'ocupada') (String).

Esta tabla tiene una relación de uno a muchos (1:*) con la tabla **Comanda**, ya que una mesa puede tener muchas comandas asociadas.

Tabla Comanda

La tabla **Comanda** registra la información de las comandas realizadas. Está compuesta por los siguientes atributos:

- **Fecha**: la fecha de la comanda (Date).
- **NumeroComensales**: el número de comensales sentados en la mesa (Integer).
- **HoraInicio**: la hora de inicio de la comanda (Time).
- **HoraFin**: la hora de finalización de la comanda (Time).
- **TotalCuenta**: el total de la cuenta (Float).

Esta tabla tiene una relación de uno a muchos (1:*) con la tabla **LineaComanda**, ya que una comanda puede tener muchas líneas de comanda asociadas.

Tabla LineaComanda

La tabla **LineaComanda** detalla los artículos pedidos en cada comanda. Está compuesta por los siguientes atributos:

- **Cantidad**: la cantidad del artículo pedido (Integer).
- **PrecioTotal**: el precio total de la línea de comanda (Float).
- **Servido**: indica si el pedido ha sido servido o no (Boolean).

Esta tabla tiene una relación de muchos a uno (*:1) con la tabla **Artículo**, ya que cada línea de comanda se refiere a un artículo específico.

Tabla Artículo

La tabla **Artículo** contiene la información de los artículos disponibles en el restaurante. Está compuesta por los siguientes atributos:

- **Nombre**: el nombre del artículo (String).
- **Precio**: el precio del artículo (Float).
- **Familia**: la familia a la que pertenece el artículo (String).

Esta tabla tiene una relación de muchos a uno (*:1) con la tabla **Familia**, ya que cada artículo pertenece a una familia específica.

Tabla ProductoManufacturado

La tabla **ProductoManufacturado** hereda de la tabla **Artículo** y representa los productos manufacturados disponibles en el restaurante. Está compuesta por los siguientes atributos:

- **Fabricante**: el fabricante del producto (String).
- **Coste**: el coste del producto (Float).

Tabla PlatoCocinado

La tabla **PlatoCocinado** hereda de la tabla **Artículo** y representa los platos cocinados disponibles en el restaurante. Está compuesta por los siguientes atributos:

- **CantidadIngrediente**: la cantidad de ingredientes utilizados en el plato (Integer).

Esta tabla tiene una relación de uno a muchos (1:*) con la tabla **Ingrediente**, ya que un plato cocinado puede contener muchos ingredientes.

Tabla Familia

La tabla **Familia** clasifica los artículos en diferentes categorías. Está compuesta por los siguientes atributos:

- **Nombre**: el nombre de la familia (String).
- **Descripción**: la descripción de la familia (String).

Tabla Ingrediente

La tabla **Ingrediente** contiene la información de los ingredientes utilizados en los platos cocinados. Está compuesta por los siguientes atributos:

- **Nombre**: el nombre del ingrediente (String).
- **CantidadStock**: la cantidad en stock del ingrediente (Float).
- **UnidadMedida**: la unidad de medida del ingrediente (String).
- **Coste**: el coste del ingrediente (Float).
- **FechaAdquisición**: la fecha de adquisición del ingrediente (Date).
- **FechaCaducidad**: la fecha de caducidad del ingrediente (Date).
- **Proveedor**: el proveedor del ingrediente (String).
- **Categoría**: la categoría del ingrediente (String).
- **Alérgenos**: los alérgenos del ingrediente (String).

Esta tabla tiene una relación de muchos a uno (*:1) con la tabla **PlatoCocinado**, ya que un ingrediente puede estar en muchos platos cocinados.

En la figura 5.2 se visualiza en forma de diagrama todas las entidades y sus relaciones, junto con sus claves primarias, de identidad, junto con sus atributos.

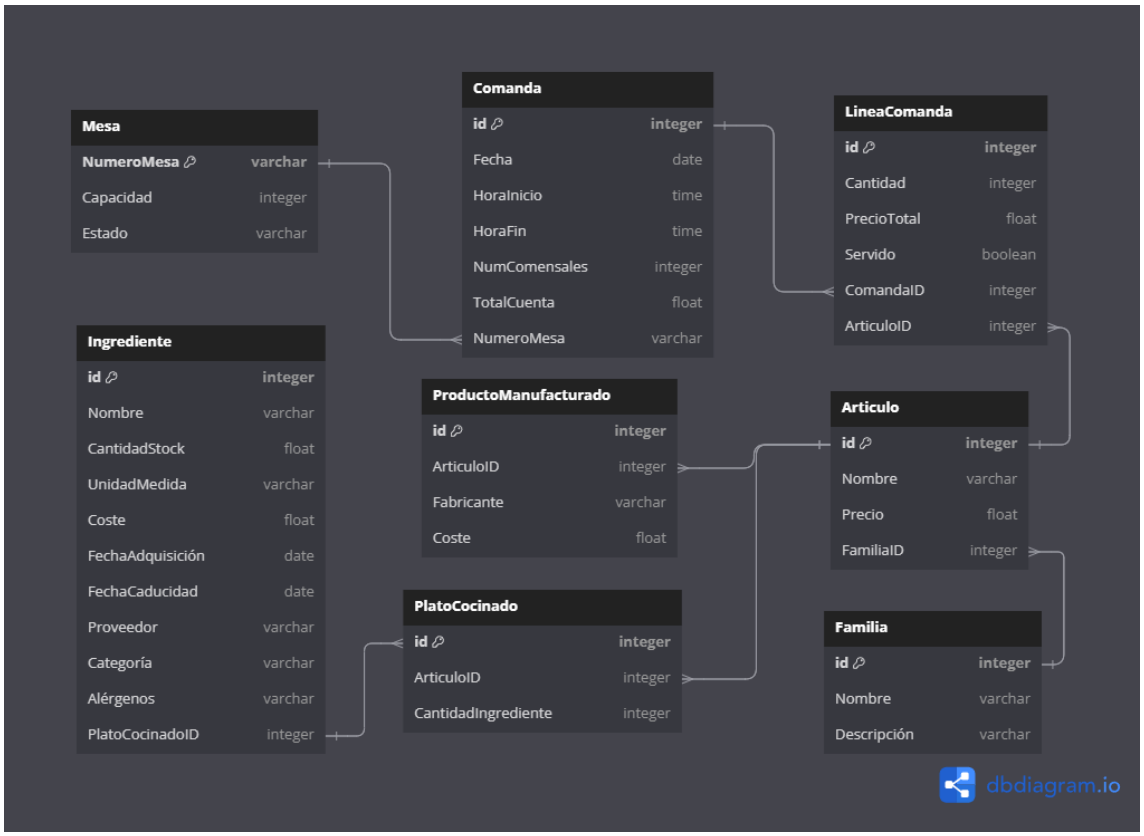

Figura 5.2: Diagrama Entidad-Relación de la BD de la aplicación desarrollada.

5.3 Bocetos de las interfaces de la aplicación

Estos son los bocetos, pantallas prototipo o mockups que se han diseñado con un estilo básico y que servirán de guía para el desarrollo de la aplicación web. En la figura 5.3 tenemos el boceto de cómo puede ser el panel de inicio para los camareros. Esta pantalla que solo verá un empleado con el rol camarero le permite navegar a la pantalla para dar de alta nuevas comandas, al listado de productos o platos que ya están listos para servir a las mesas, y a consultar las mesas ocupadas que tienen una comanda activa.

En la figura 5.4 vemos un menú prototipo para que los camareros tomen nota de la comanda en cada mesa. Tienen estructurados los productos que vende el local por familias: bebidas, entrantes, platos y postres. A la derecha se listan los ítems de la comanda y un botón para registrarla.

La figura 5.5 muestra una lista con todos los ítems que se han confirmado en una comanda, y que están listos para servirse. Cuando un camarero va a servirlos pulsa en el botón Servir, y se le asigna el ítem, que queda marcado como servido para que no haya duplicidad en las entregas con otros camareros.

El terminal de cocina que se aprecia en la figura 5.6 muestra una lista similar a la anterior, pero a esta pantalla solo accede el personal de cocina. Es una lista de todos los platos que deben ser preparados en la cocina, llegan por orden de comanda, y el encargado de cocina solo debe pulsar el botón Listo y esto manda un aviso a la pantalla del servicio a mesa, indicando al camarero que ya puede servir los platos.

La figura 5.7 muestra una interfaz a la que puede acceder tanto el camarero como el gerente, pero solo el gerente tiene la opción de calcular la cuenta. Pulsando el botón Calcular cuenta aparece la suma de todo lo que tienen que pagar los comensales y la comanda de esa mesa en particular se queda cerrada, dejando la mesa disponible.

En la figura 5.8 vemos la pantalla para dar de alta artículos nuevos. El gerente es el único que tiene acceso a esta pantalla, y en ella puede dar de alta artículos, recetas de los platos que se cocinan en el restaurante e ingredientes. A esta pantalla se accede desde el menú de administrador de la figura 5.9, quien además puede consultar las mesas para generar la cuenta y consultar las estadísticas de su restaurante.

Figura 5.3: Prototipo de pantalla del panel de inicio para los camareros.

Figura 5.4: Prototipo de pantalla para registrar una comanda.

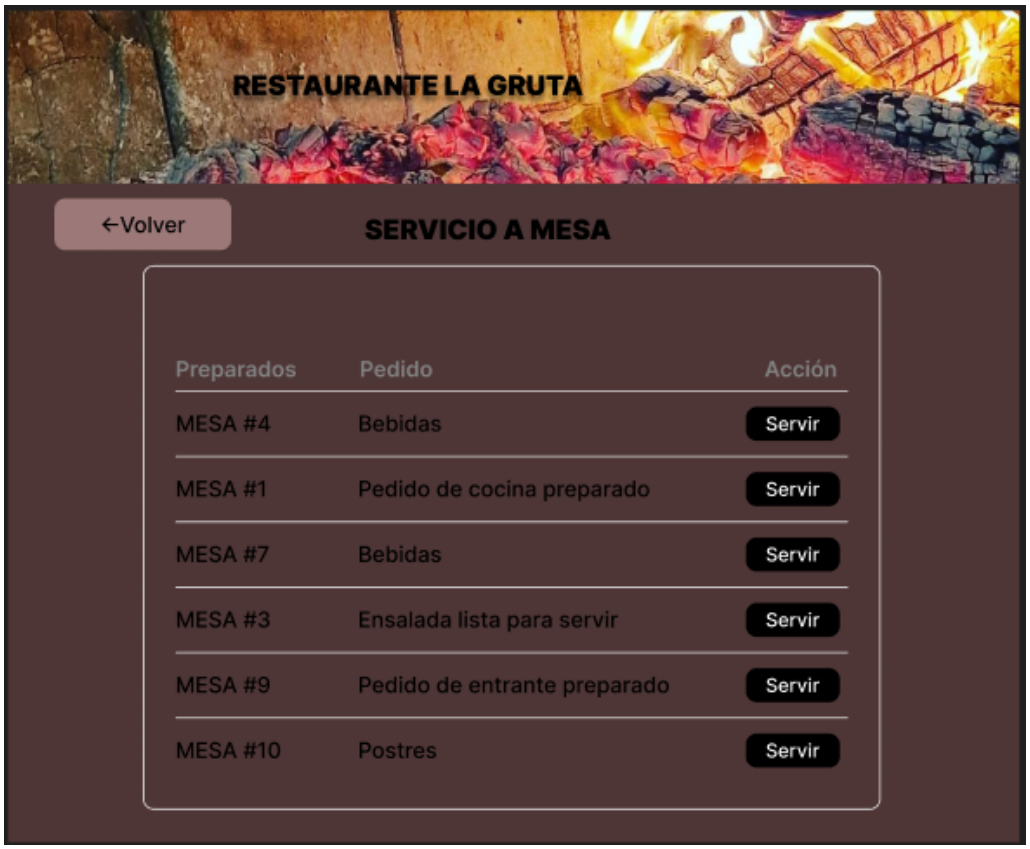

Figura 5.5: Prototipo de pantalla con la lista de artículos pendientes para servir.

Figura 5.6: Prototipo pantalla con la lista de pedidos pendientes para cocina.

Figura 5.7: Prototipo de pantalla para consultar las mesas.

Figura 5.8: Prototipo de pantalla de gestión de artículos.

Figura 5.9: Prototipo de pantalla del panel del administrador

CAPÍTULO 6
Desarrollo de la solución

En esta sección se describen los pasos clave que se han seguido para desarrollar la aplicación, comenzando por la arquitectura del sistema desarrollado y la tecnología empleada. Todos estos elementos han colaborado para construir un entorno de desarrollo sólido, y gracias a la interoperabilidad entre ellos se ha creado una simbiosis de tecnologías que ha hecho posible construir la aplicación que se documenta en el capítulo 7.

6.1 Arquitectura

Después de un primer proceso de análisis, especificación de requisitos y diseño, la aplicación ha pasado por varias fases de desarrollo. Primero se preparó el entorno de desarrollo instalando las distintas tecnologías para trabajar con el frontend, después se levantó una API para el backend para ir consolidando la persistencia de los datos en cada nueva pantalla.

La aplicación web ha sido desarrollada utilizando el framework React [Rea24] [Pel19] junto con Vite [Vit24], Redux [Red24] y Cypress [Cyp24] para la construcción, consolidación y validación del proyecto.La aplicación permite a los usuarios interactuar con el sistema mediante una navegación basada en menús con botoneras y realizar funcionalidades básicas para gestionar una carta digital de un restaurante. La comunicación entre la aplicación web y el servidor backend se realiza a través de peticiones HTTP utilizando la librería Axios [Axi24]. Por otro lado, las peticiones y respuestas entre el frontend y el backend se intercambian en formato JSON [JSO24].

El servidor backend está implementado con Node.js [Nod24] y Express [Exp24], con el que se ha construido una API RESTful que sirve todas las solicitudes del frontend [Red23]. Las principales responsabilidades del servidor backend incluyen la gestión de la lógica de negocio, la conexión y manipulación de la base de datos y la creación de endpoints para las operaciones CRUD, es decir, la lectura, el almacenamiento, la actualización y el borrado de la información de nuestra base de datos.

El servidor backend recibe las solicitudes del frontend, realiza las operaciones necesarias en la base de datos y devuelve las respuestas apropiadas. Todo el intercambio de datos entre el frontend y el backend se realiza en formato JSON, facilitando la integración y el manejo de la información.

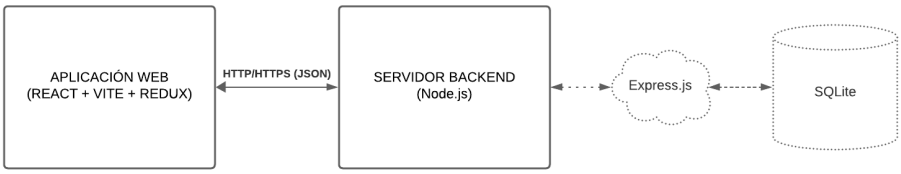

Figura 6.1: Arquitectura de la aplicación desarrollada.

La base de datos utilizada es SQLite [SQL24], y se ha elegido por lo ligera que es con proyectos de la escala que estamos desarrollando. La base de datos almacena toda la información necesaria para el funcionamiento del sistema, como ingredientes, recetas, o comandas. La estructura de la base de datos se define mediante tablas relacionadas, y las consultas SQL se utilizan para realizar operaciones CRUD sobre las tablas. En la figura 6.1 se puede ver un diagrama de arquitectura básico del sistema.

6.2 Contexto tecnológico

En esta sección se ofrece un breve resumen de todas las tecnologías utilizadas que, combinadas, han permitido que la aplicación pueda funcionar. Se detallan sus características y la justificación de por qué se han empleado en este proyecto.

6.2.1. Lenguajes de programación

Como lenguaje de programación se ha utilizado **JavaScript** [Jav24a] tanto para el *frontend* como para el *backend* del proyecto. Es un lenguaje de programación interpretado que se emplea en muchos ámbitos, especialmente el de desarrollo de aplicaciones; y según una encuesta y un estudio del *State of JS* [Est22], es el lenguaje líder de uso en React. JavaScript es versátil y tiene un extenso ecosistema de bibliotecas y *frameworks* que lo han convertido en una opción atractiva para los desarrolladores de páginas webs modernas.

6.2.2. Frameworks, bibliotecas y tecnologías

ReactJS es un *framework* de JavaScript para construir interfaces de usuario que permite desarrollar componentes reutilizables y manejar de manera eficiente el estado de la aplicación. Su capacidad para crear aplicaciones de una sola página SPA (Single-page Application) [Dig24] con un rendimiento muy superior y optimizado es una de las razones principales por las que fue elegido para trabajar en este proyecto. Gracias a su flujo de datos unidireccional se facilita la gestión y el seguimiento del estado de la aplicación.

Redux es una biblioteca de JavaScript para la gestión centralizada del estado de la aplicación, que proporciona una única fuente de verdad, es decir, el repositorio donde se encuentran los datos más precisos de una aplicación. Esto facilita la gestión y el mantenimiento del estado de una manera predecible, lo que ayuda a prevenir errores que, de otro modo, serían muy difíciles de depurar. Además, tiene soporte para *middlewares* para manejar tareas asíncronas.

Express.js es un *framework* de Node.js para la creación de aplicaciones web y de APIs, proporcionando una estructura minimalista y flexible para el desarrollo del *backend*. Esta herramienta ha permitido la creación de la API RESTful de manera rápida y eficiente, lo

que le ha supuesto que la comunicación entre el *front* y el *back* haya sido fluida. Tiene un amplio soporte para *middlewares* que facilitan la gestión de solicitudes y respuestas.

SQLite es un sistema gestor de bases de datos relacionales basado en una biblioteca de C que proporciona una base de datos SQL ligera y autocontenida. Es especialmente útil para aplicaciones que requieren un almacenamiento de datos local y no necesitan un servidor de bases de datos completo. Ofrece un entorno de integridad y fiabilidad con todas las características ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) [Ara23] para asegurar una transferencia de datos exitosa.

Axios es un cliente HTTP basado en promesas para realizar solicitudes desde el navegador y Node.js, que se utiliza en el *frontend* para interactuar con la API del *backend*, facilitando en gran medida las operaciones de lectura, creación, edición y borrado. Sus puntos fuertes son una interfaz sencilla y el manejo eficiente de operaciones asíncronas gracias a su soporte para promesas.

CORS, de sus siglas en inglés Cross-Origin Resource Sharing, es un *middleware* para Express.js que permite el paso cruzado de solicitudes. Este es necesario para permitir que el *frontend* interactúe con el *backend* cuando cada uno está alojado en distintos dominios. Gracias a su seguridad y flexibilidad se puede controlar qué dominios pueden acceder a los recursos del servidor.

Vite es una herramienta de construcción del *frontend* de las aplicaciones web, que proporciona un entorno de desarrollo ágil y optimizado para la creación de aplicaciones modernas. Vite permite una recarga en caliente eficiente y una experiencia de desarrollo mejorada. Se trata de una de las herramientas de desarrollo más potente con las que se ha trabajado para realizar este proyecto, ya que con solo guardar los cambios en el código estos se veían reflejados inmediatamente en la aplicación web sin aplicar ningún tipo de recarga.

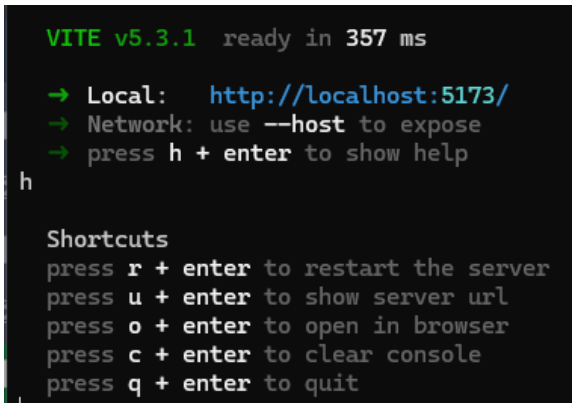

Figura 6.2: Ventana de terminal del sistema que muestra el inicio del cliente web con VITE y la dirección local para visualizar la APP.

Node.js es el entorno de ejecución de JavaScript del lado del servidor, que interpreta todo el código facilitando el desarrollo de aplicaciones web completas con un único lenguaje de programación. Sus puntos fuertes es que maneja eficientemente las operaciones de entra y de salida, y que tiene un sistema de paquetes y librerías muy extenso que facilita la gestión de las dependencias.

Charts.js es una biblioteca famosa de JavaScrip para visualizar los datos en forma de gráficos [Cha24]. Es una herramienta sencilla y flexible, que permite la creación de gráficos interactivos. Mediante una API de la base de datos se puede transmitir la información y pintar las gráficas en un componente de la aplicación.

WebSockets es una tecnología avanzada que permite abrir un canal de comunicación interactiva entre el navegador del cliente y un servidor [Jav24b]. Gracias a esta tecnología se pueden enviar actualizaciones en tiempo real desde el servidor al cliente sin necesidad de que este solicite los datos activamente. El *frontend* y el *backend* mantienen una relación de publicador-suscriptor y están informados de lo que ocurre en todo momento.

React Toastify es una biblioteca para React que permite mostrar notificaciones personalizables en forma de mensajes emergentes, ofreciendo una retroalimentación rápida y eficiente de lo que está sucediendo en el sistema de un modo no intrusivo para el usuario.

Nodemon es una herramienta que ayuda a desarrollar aplicaciones basadas en Node.js y su cometido es reiniciar automáticamente la aplicación del servidor cuando se detectan cambios en el *script* de configuración. Este reinicio automático permite a los desarrolladores ver los efectos de sus cambios de inmediato sin necesidad de reiniciar manualmente el servidor, lo que les permite seguir trabajando en el desarrollo de su aplicación sin estar pendientes del estado del servidor. En la figura 6.3 se observa una ventana de terminal del sistema con Nodemon iniciando el servidor, y cómo se ha reiniciado indicando fecha y hora del reinicio.

```
C:\Users\jlsor\gestor-lagruta\backend>nodemon index.js
[nodemon] 3.1.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server running at http://localhost:3000/ - 27/6/2024, 7:23:25
Connected to the SQLite database.
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server running at http://localhost:3000/ - 27/6/2024, 7:38:44
Connected to the SQLite database.
```

Figura 6.3: Inicio de servidor con Nodemon, y su reinicio al detectar un cambio en la configuración.

6.2.3. Justificación de uso

La elección de estas herramientas y los *frameworks* junto con JavaScript como lenguaje de programación, se fundamenta en la capacidad conjunta de estos para satisfacer los requisitos del proyecto de manera eficiente y escalable, ya que en un restaurante en hora punta entran muchos pedidos y el flujo de información es enorme; y en una aplicación como esta, que la información se maneje rápida y fiablemente es un requisito indiscutible.

Para ello, ReactJS y Redux proporcionan una base sólida para el desarrollo de una interfaz de usuario interactiva y de fácil mantenimiento. Express.js y SQLite, por su parte, ofrecen una solución de *backend* ligera y eficiente para la gestión de datos y la creación de APIs. El uso de Axios y CORS facilita la comunicación entre el *frontend* y el *backend*, asegurando una integración fluida y segura. Además, herramientas como Vite y Node.js mejoran significativamente la experiencia de desarrollo y despliegue, permitiendo un flujo de trabajo ágil y optimizado.

Por otro lado, tecnologías como los WebSockets permiten que las pantallas de las terminales se actualicen en tiempo real sin necesidad de refrescar los componentes manualmente. Esto dota a la aplicación de un nivel de dinamismo muy elevado y la convierte en una solución muy interesante para que cualquier negocio de restauración y para su implementación en cualquier tipo de entorno. Finalmente, React Toastify ha permitido que podamos notificar al camarero que ha añadido con éxito un ítem de la carta a la comanda, y también que podamos mostrar otras notificaciones cuando se marca un plato como preparado o como servido.

6.2.4. Herramientas de desarrollo

Además de las herramientas que han ayudado a darle vida al proyecto desde dentro, también se han utilizado las siguientes herramientas que han contribuido para dársela desde fuera.

Vistual Studio Code es un editor de código fuente desarrollado por Microsoft y una herramienta potente que facilita el trabajo de desarrollo de aplicaciones de cualquier tipo. Se le pueden instalar una gran variedad de extensiones para añadir funcionalidades adicionales al editor, y lleva herramientas de depuración integradas para JavaScript y para otros lenguajes. Gracias a su integración con Git se ha podido llevar un control de versiones y la confianza de que todo el proyecto estaba a salvo de imprevistos.

BD Broswer for SQLite es una herramienta de visualización y edición de bases de datos SQLite que ha proporcionado una interfaz gráfica intuitiva para comprobar que la API manipulaba la base de datos correctamente. Con esta herramienta se pueden ver y editar las tablas de la BD, hacer consultas SQL directamente desde la interfaz e insertado y borrado de datos.

Figma es una aplicación web con la que se pueden diseñar interfaces y realizar un prototipado colaborativo en tiempo real. Los bocetos de la aplicación se han diseñado con Figma, proceso que ha sido muy sencillo gracias a la reutilización de componentes y a la definición de estilos globales.

Lucidchart es una herramienta para realizar todo tipo de diagramas que permite la colaboración en tiempo real con otros usuarios. Dispone de numerosas plantillas para todo tipo de diagramas y se puede integrar a herramientas como Google Drive, Slack o Microsoft Office.

6.3 Ejemplos de código

En esta sección detallaremos los fragmentos de código más interesantes y sobre los que se apoya el proyecto, y una breve descripción de su funcionamiento.

6.3.1. Código del servidor

El trozo de código de la figura 6.4 muestra cómo se ha creado la conexión a la base de datos que por defecto tiene el nombre de fichero *my-database.db*. Tiene un control de errores por si la conexión falla por algún motivo, y en las últimas líneas se puede observar

que se ha utilizado el método *serialize()*. Este método sirve para crear las tablas iniciales y se puede eliminar del código en futuros arranques del servidor.

```
// Conexión a la base de datos SQLite
const db = new sqlite3.Database('./my-database.db', (err) => {
  if (err) {
    console.error('Error opening database ' + err.message);
  } else {
    console.log('Connected to the SQLite database.');

    // Creación de tablas
    db.serialize(() => {
      db.run(`CREATE TABLE IF NOT EXISTS Mesa (
        NumeroMesa TEXT PRIMARY KEY,
        Capacidad INTEGER,
        Estado TEXT
      )`);
```

Figura 6.4: Función que conecta a la base de datos del sistema y crea las tablas de la aplicación.

En la siguiente figura 6.5 se detalla una pequeña función que se programó para saber la fecha y la hora en la que se inicia o se reinicia el servidor con Nodemon cuando este detecta cambios en el script de configuración. El resultado de este código puede apreciarse en la figura 6.3.

```
// Iniciar el servidor
server.listen(port, () => {
  const now = new Date().toLocaleString();
  console.log(`Server running at http://localhost:${port}/ - ${now}`);
});
```

Figura 6.5: Función que indica la fecha y hora del servidor cuando se inicia o se reinicia.

En la figura 6.6 está detallada la función con la que el servidor mantiene informados a todos los clientes del *frontend* mediante la propagación de señales *broadcast*. Gracias a estas señales, los distintos componentes del sistema cuya interfaz requiere una actualización constante de datos mostraba información actualizada del estado de la aplicación.

El siguiente ejemplo de la figura 6.7 se muestra una función que hace una llamada a la API del servidor. Esta llamada crea un listado con todos los ingredientes registrados en la base de datos del sistema. El resto de llamadas a la API hacen llamadas similares a esta, y las consultas SQL se han escrito teniendo en cuenta las distintas necesidades para que el *frontend* pudiera hacer operaciones CRUD sobre los datos. La figura 6.8 es un recorte de la ventana del navegador donde se puede ver una entrada de la tabla Ingrediente en formato JSON cuando se hace una llamada a la API con el código de la figura anterior.

```
// Función para mandar un Broadcast a todos los clientes conectados
const broadcast = (data) => {
  wss.clients.forEach(client => {
    if (client.readyState === WebSocket.OPEN) {
      client.send(JSON.stringify(data));
    }
  });
};
```

Figura 6.6: Función para mandar señales broadcast con WebSockets.

```
// Llamada a la api para mostrar listar los ingredientes
app.get('/api/ingredientes', (req, res) => {
  db.all('SELECT * FROM Ingrediente', [], (err, rows) => {
    if (err) {
      console.error('Error fetching ingredientes: ', err);
      res.status(400).json({ error: err.message });
      return;
    }
    res.json(rows);
  });
});
```

Figura 6.7: Función de la API del servidor para listar los ingredientes.

El último fragmento de código destacable del servidor es esta llamada a la API que se observa en la figura 6.9. Tiene dos funciones, la primera es marcar como Preparado un plato cuando el chef del restaurante ha indicado desde la aplicación que el plato ya está listo para ser servido. Y su segunda función es crítica para el funcionamiento óptimo de la aplicación, ya que manda una señal *broadcast* del tipo `UPDATE_PREPARADO` informando de que hay un pedido preparado. Esta señal se recibe en todos los *sockets* de escucha, que mandan la señal de actualizar los datos que se muestran en la pantalla del terminal de cocina, la de servicio a mesas y en la de consultar mesas.

6.3.2. Código de la aplicación

En la figura 6.10 tenemos un ejemplo básico de cómo se crea la lógica de negocio de una interfaz. Se trata de RegistrarComanda, el componente más complejo de toda la aplicación y que reúne muchas funcionalidades programadas de forma similar en otros componentes. En esta sección nos centraremos en explicar el funcionamiento de este componente.

Cada interfaz comienza con una función con su nombre, y en ella se declaran los métodos y las variables o constantes. En este ejemplo declaramos las mesas disponibles,

Figura 6.8: Llamada a la API de ingredientes desde un navegador web.

```
// Marcamos los productos como cocinados
app.put('/api/lineas/:id/preparar', (req, res) => {
  const id = req.params.id;
  const query = `UPDATE LineaComanda SET Preparado = 1 WHERE id = ?`;

  db.run(query, [id], function(err) {
    if (err) {
      res.status(400).json({ error: err.message });
      return;
    }
    // Broadcast para notificar de la actualización a los clientes
    broadcast({ type: 'UPDATE_PREPARADO', id });
    res.json({ message: 'Línea de comanda marcada como preparada', updatedID: id });

  });
});
```

Figura 6.9: Función de la API para marcar un producto como Preparado por el cocinero.

la lista de productos de una familia, la mesa que seleccionará el camarero para iniciar o editar una comanda, la comanda activa para la mesa seleccionada, si la tiene, todos los productos que forman la comanda, y el WebSocket para recibir las actualizaciones en tiempo real. Debajo está la función de navegación para el botón Atrás, y nos lleva a la ruta especificada, en este caso para volver al Panel de Camareros.

Después está el método useEffect, que es un gancho que se ejecuta al montar el componente cuando selectedComanda cambia. La función fetchMesas se utiliza para obtener las mesas disponibles, y el WebSocket se deja escuchando los mensajes en tiempo real, y si recibe un mensaje de tipo `NEW_LINEA_COMANDA` actualiza las líneas de la comanda.

Las funciones asíncronas del componente son fecthMesas, para obtener una lista de mesas; fetchProductos, para obtener los productos de una familia en específico; fetchComandaActiva, que obtiene la comanda que esté activa para una mesa en particular y si no hay comanda, crea una nueva; y fetchLineasComanda, que obtiene todas las líneas de pedido de una comanda específica.

En lo que respecta al manejo de eventos, handleMesaChange se encarga de cambiar la mesa seleccionada y obtener la comanda activa para esa mesa; handleFamiliaClick obtiene los productos de una familia en particular cuando se hace clic en un botón de familia; y handleAddProducto añade un producto a la comanda, pero si ya existe una línea para ese producto se suma la cantidad, y si no, se crea una línea nueva.

Después se renderiza toda la información en pantalla, y se mezcla código HTML con código JavaScript para lograrlo. Primero se pinta un combo para seleccionar la mesa, después aparecen una lista de productos de cada familia (bebidas, postres, platos, entrantes) y al hacer clic en ellos se despliega una lista con todos los productos registrados en el sistema. Como último elemento, una tabla con todos los productos que se han ido añadiendo a la aplicación. Finalmente, como elemento oculto a la interfaz inicial, la notificación con Toastify que solo saldrá cuando se añada un producto a la comanda.

```
const API_URL = 'http://localhost:3000/api';

function RegistrarComanda() {
  const [mesas, setMesas] = useState([]);
  const [productos, setProductos] = useState([]);
  const [selectedMesa, setSelectedMesa] = useState('');
  const [selectedComanda, setSelectedComanda] = useState(null);
  const [lineasComanda, setLineasComanda] = useState([]);
  const [ws, setWs] = useState(null);
  const navigate = useNavigate();

  const atras = () => {
    navigate('/panel-camareros');
  };

  useEffect(() => {
    fetchMesas();
    const socket = new WebSocket('ws://localhost:3000');
    setWs(socket);
    socket.onmessage = (event) => {
      const message = JSON.parse(event.data);
      if (message.type === 'NEW_LINEA_COMANDA') {
        fetchLineasComanda(selectedComanda.id);
      }
    };
    return () => socket.close();
  }, [selectedComanda]);
```

Figura 6.10: Función de creación de la pantalla Registrar Comanda.

CAPÍTULO 7
Producto desarrollado

Este capítulo está dedicado a explicar el funcionamiento de la aplicación y a documentar todas sus pantallas y características. Es importante destacar que esta aplicación no está completa, sino que se trata de un MVP o mínimo producto viable [Uni24c]. Los MVP son versiones de software que tienen todas sus funcionalidades accesorias pendientes de desarrollo, y están limitadas a las funcionalidades que son los puntos fuertes de la aplicación. Con esto se pretende medir el interés de los clientes potenciales y atraer la atención de estos hacia nuestro producto sin invertir muchos recursos en el software. Lo que se gana con esto es ahorrar tiempo, dinero y esfuerzo, por lo que seguir una estrategia MVP ha sido la filosofía preferida de muchas startups existosas.

7.1 Interfaces de la aplicación

7.1.1. Pantalla de inicio

El la figura 7.1 tenemos la pantalla de inicio y cabecera de la aplicación, que se repite en todas las pantallas y que simula el acceso al terminal de Camareros, Cocina y Gerente mediante botones. A esas secciones solo se podría llegar teniendo el rol de usuario correspondiente en la versión final del producto, y serían inaccesibles para cualquiera que no tuviera acceso. No obstante, para facilitar el proceso de demostración del producto a clientes potenciales, y especialmente porque se trata de un MVP, no se ha desarrollado la gestión de usuarios y roles en esta versión.

Figura 7.1: Pantalla de inicio de la aplicación, con los acceso de los 3 actores de la aplicación.

7.1.2. Panel de camareros

Al hacer clic en Camareros accedemos a un panel diseñado para este rol, como se ve en la figura 7.2, con tres opciones: Registrar Comanda, Servicio a Mesa y Consultar Mesa.

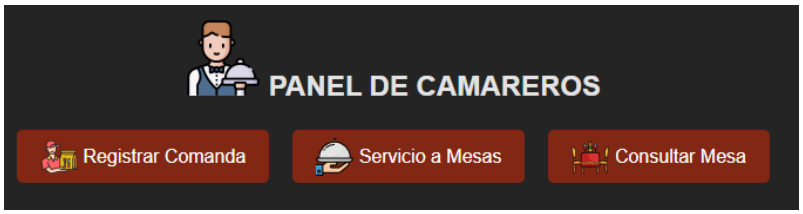

Figura 7.2: Pantalla con el panel de los camareros y el acceso a todo lo que estos pueden hacer en la aplicación.

Registrar Comanda

En registrar comanda hay un desplegable con todas las mesas que tiene el local, y el camarero tiene que empezar el proceso de registro de comanda seleccionando una mesa. En la figura 7.3 se observa que el sistema actualmente tiene hasta la mesa 6 registrada. Una vez seleccionada la mesa aparece una lista con botones según las familias dadas de alta en el sistema, y el camarero puede ir añadiendo ítems de la carta a la comanda, tal y como se puede ver en la figura 7.4. Una vez se pulsa en el botón Añadir, si no ha habido problemas sale un mensaje de éxito, como se ve en la figura 7.5. Como último elemento de esta pantalla, tal y como puede verse en la figura 7.6, el camarero puede hacer un seguimiento rápido de todo lo que hay en la comanda, por si necesita recordar a los comensales lo que se acaba de pedir.

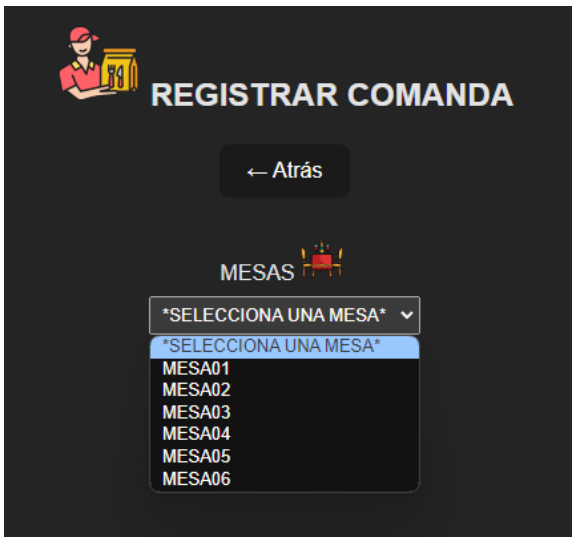

Figura 7.3: Pantalla para registrar o editar una comanda asignada a una mesa.

Servicio a Mesas

La sección de Servicio a Mesas de la figura 7.7 muestra una lista de todos los ítems de la comanda que están listos para ser servidos. Esta lisa filtra cualquier tipo de plato cocinado que ya ha sido marcado como Preparado, y cualquier bebida o producto de la carta que ya está listo para servir. El camarero sabe qué producto y a qué mesa tiene que llevarlo, y cuando los lleva los marca como servidos pulsando un botón y confirmando

Figura 7.4: Menú con las familias de artículos y los ítems registrados en el sistema para la familia Platos.

Figura 7.5: Mensaje emergente de que el artículo ha sido añadido con éxito a la comanda.

el mensaje, y el ítem desaparece de la lista. Es importante destacar que los productos se muestran por orden en el que fueron pedidos, por lo que siempre será prioritario llevar los platos o las bebidas de quien pidió primero.

Consultar Mesas

En esta pantalla de Consultar Mesas se muestran todas las mesas con comandas activas, y en de la figura 7.8 vemos una de ellas, la mesa con el identificador MESA06. Esta pantalla se actualiza en tiempo real nada más hay un cambio en el estado de uno de los ítems de la comanda, donde se puede saber de un vistazo si el pedido está servido, o si aún sigue en preparación en cocina.

7.1.3. Terminal de Cocina

Esta es la única pantalla a la que tienen acceso los cocineros, en ella pueden ver una lista de todos los pedidos pendientes, que les salen por orden en el que fueron pedidos, teniendo más prioridad los primeros platos del listado. En la figura 7.9 se observa la lista, y cómo al pulsar el botón de Preparado sale un mensaje de confirmación. Al confirmarlo

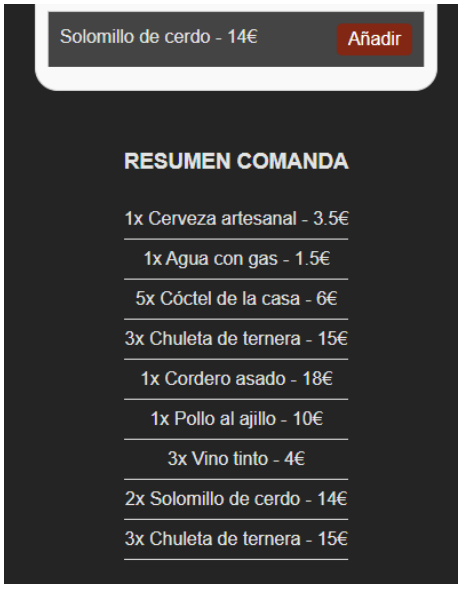

Figura 7.6: Resumen de la comanda para mantener informado al camarero.

Figura 7.7: Pantalla del Servicio a Mesas con una lista de todo lo que hay pendiente de servir.

el ítem desaparece de la lista y se muestra como Preparado en la tabla de la mesa en Consultar Mesas.

7.1.4. Panel de administrador

El panel de administrador de la figura 7.10 tiene tres botones que enlazan a otras secciones de la aplicación: la Gestión de Artículos, donde se podrá dar de alta ingredientes, añadir ingredientes a las recetas y comprobar el stock de productos comprados a otros proveedores; Consultar Mesa, que lleva a la misma pantalla a la que puede acceder el Camarero para ver los estados de las mesas y las comandas; y el panel de estadísticas, donde se verán distintas métricas del rendimiento de su restaurante. De todas estas secciones, en primer lugar se detallan las del panel de Gestión de Artículos de la figura 7.11.

Figura 7.8: Pantalla de mesas con todos los detalles de la comanda.

Ingredientes

En esta pantalla el administrador puede dar de alta un inventario de todos los ingredientes del restaurante, y puede almacenar datos como el nombre, el stock, el coste, el proveedor, su categoría (vegana, etc), los alérgenos y las fechas de compra y de caducidad. En la figura 7.12 se observa el formulario de carga de datos y la tabla de ingredientes, que se pueden editar y borrar. Al pulsar en editar los campos de Añadir Ingrediente se transforman automáticamente en Editar Ingrediente, y al guardar los datos vuelve el formulario de ingrediente nuevo. La tabla se comporta como una tabla de datos, y permite ordenar por fecha de caducidad para poder saber en cualquier momento qué ingredientes del inventario están a punto de caducar. Esta funcionalidad es de vital importancia para evitar el desperdicio alimentario en la hostelería, ya que el gerente puede ver de un vistazo qué ingredientes quedan en el local que están próximos a la máxima fecha de consumo recomendada.

Platos Caseros

En esta pantalla el gerente puede visualizar una lista de platos caseros y puede añadir ingredientes, tal y como se comprueba en la figura 7.13. Y en la pantalla de la figura 7.14 se puede ver una lista del inventario de productos comprados a otros proveedores. Es importante recalcar que el diseño de Ingredientes, Platos Caseros y Productos Comprados difiere mucho entre ellos de manera intencionada, ya que se van a utilizar para recoger feedback por parte del cliente.

Figura 7.9: Terminal de cocina exclusivo para usuarios con este rol, muestra una lista de pedidos de cocina activos.

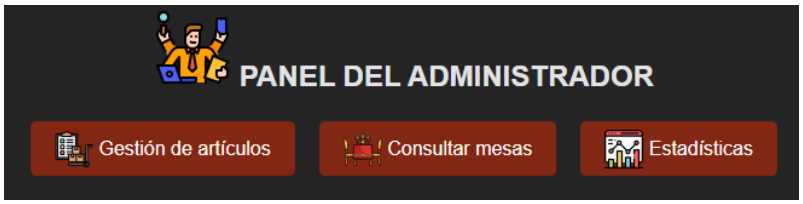

Figura 7.10: Panel del administrador con los enlaces a otras secciones de la app.

Estadísticas

La última pantalla de la aplicación es la de Estadísticas, que gracias a la librería de Chart.js nos ha permitido representar mediante gráficas varios indicadores y métricas con la información de la que disponemos en la base de datos. En la figura 7.15 observamos los productos más demandados en el restaurante.

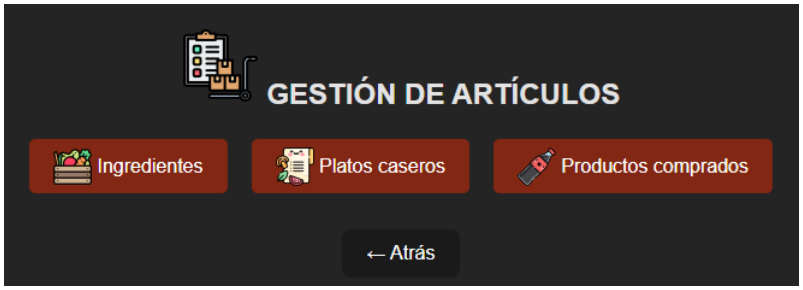

Figura 7.11: Submenú para la gestión de artículos.

Figura 7.12: Formulario de carga para hacer inventario de ingredientes.

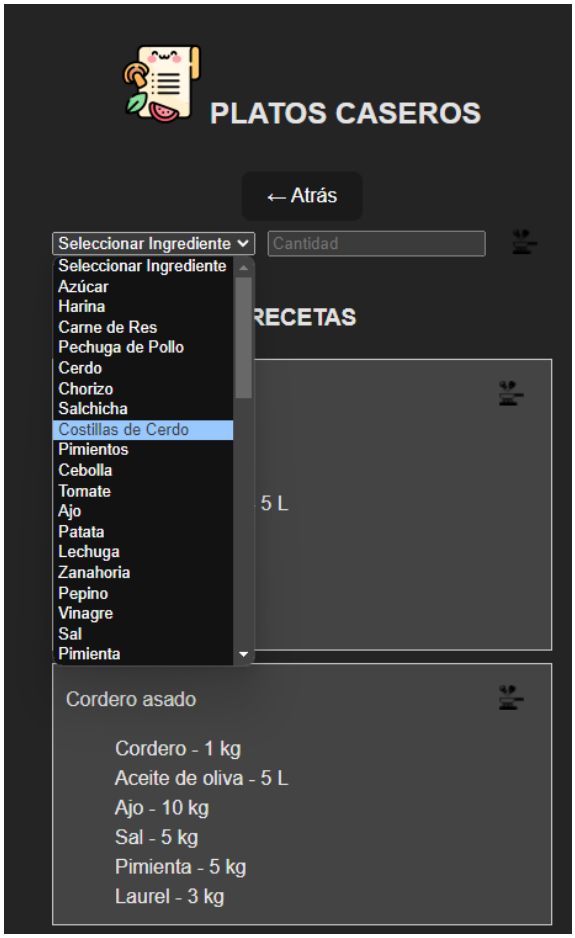

Figura 7.13: Pantalla de platos caseros para editar las recetas del restaurante.

Figura 7.14: Pantalla con el inventario de los productos comprados a proveedores.

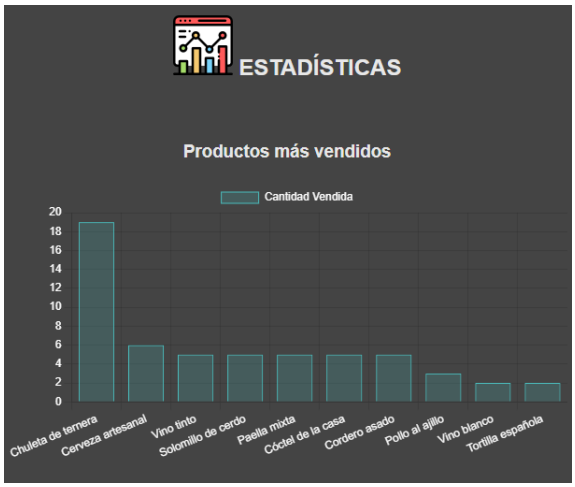

Figura 7.15: Pantalla de estadísticas que muestran los productos más vendidos.

CAPÍTULO 8
Validación

Una vez tenemos desarrollada la aplicación, la validación y la verificación son componentes críticos que nos ayudan a asegurar la funcionalidad y la calidad del producto final. A medida que un proyecto software crece y se vuelve más complejo, la importancia de implementar pruebas automatizadas y exhaustivas se hace patente. Mediante estas pruebas podemos garantizar que el sistema funcione como esperamos y, además, nos ayuda a prevenir errores y problemas de usabilidad que podrían surgir más tarde después de su implementación.

La validación de software se aplica para confirmar que este cumple con los requisitos y las expectativas del usuario final. Este proceso abarca desde la verificación de la interfaz de usuario hasta la validación de la lógica de negocio y la integridad de la base de datos. En este proyecto vamos a realizar varias pruebas automatizadas con Cypress, una herramienta muy polivalente con la que se pueden simular interacciones de usuario y validar que todos los componentes del sistema funcionen correctamente bajo diversas condiciones. Y en otra sección, aplicaremos las 10 heurísticas de Nielsen para evaluar la experiencia del usuario.

8.1 Pruebas automáticas con Cypress

En esta sección se documenta cómo se han implementado y ejecutado varias pruebas automáticas de CRUD sobre la API de ingredientes utilizando el entorno de validación de Cypress. Pruebas como esta aseguran que las operaciones básicas de la API de la base de datos funcionan correctamente.

Para verificar que la API puede crear nuevos ingredientes correctamente se hace una prueba de creación sobre la API de ingredientes. En esta prueba se envía al servidor una solicitud POST con los datos de un nuevo ingrediente y se valida su respuesta para comprobar que el ingrediente se ha creado sin problemas. En la figura 8.1 se observa el código empleado para la validación.

En el resto de pruebas seguimos procedimientos muy similares. Por ejemplo, para asegurarnos de que la API puede leer los datos correctamente realizamos una prueba de lectura enviado una solicitud GET para obtener una lista de ingredientes, validando después que la respuesta contenga los datos que esperamos. En la figura 8.2 se puede ver el código de esta validación.

```
describe('CRUD de Ingredientes', () => {
  it('Debe crear un nuevo ingrediente', () => {
    cy.request('POST', 'http://localhost:3000/api/ingredientes', {
      nombre: 'Nuevo Ingrediente',
      cantidadStock: 10,
      unidadMedida: 'kg',
      coste: 2.5,
      fechaAdquisicion: '2024-06-15',
      fechaCaducidad: '2025-06-15',
      proveedor: 'Proveedor X',
      categoria: 'Categoria Y',
      alergenos: 'Ninguno'
    }).then((response) => {
      expect(response.status).to.equal(201);
      expect(response.body).to.have.property('id');
    });
  });
});
```

Figura 8.1: Prueba de validación en Cypress para la creación de un ingrediente.

```
describe('CRUD de Ingredientes', () => {
  it('Debe obtener la lista de ingredientes', () => {
    cy.request('GET', 'http://localhost:3000/api/ingredientes')
      .then((response) => {
        expect(response.status).to.equal(200);
        expect(response.body).to.be.an('array');
      });
  });
});
```

Figura 8.2: Prueba de validación en Cypress para la lectura de ingredientes.

Para verificar que la API actualiza los ingredientes correctamente se realiza una prueba de actualización. Para comprobar que el update a la base de datos se hace sin problema se envía una solicitud PUT para modificar los datos de un ingrediente existente, y se valida mediante un request que el ingrediente ha sido actualizado con éxito. Para la última prueba, la de eliminación, enviamos una solicitud DELETE para eliminar un ingrediente existente y validamos que la respuesta confirme su eliminación.

Una vez hemos programado las pruebas, las pasamos al entorno de validación de Cypress y comprobamos que todas las pruebas se han superado con éxito, tal y como se puede comprobar en la figura 8.3. La validación de cualquier operación sobre la base de datos y la API es un paso fundamental para garantizar la fiabilidad del sistema. Programar varias pruebas automatizadas puede ayudar a detectar y corregir errores antes de que el sistema sea utilizado en un entorno de producción.

8.2 Validación heurística de la plataforma

La validación heurística es una técnica que nos permite evaluar la usabilidad de una aplicación, y se realiza a través de un análisis sistemático de la interfaz de usuario y de su comparación con un conjunto de principios heurísticos. Para esta validación utilizaremos las 10 reglas heurísticas de Jakob Nielsen [Nie92], y como la aplicación se trata de un MVP se le pondrá una atención especial a esta validación para que futuras versiones de esta queden mucho más refinadas.

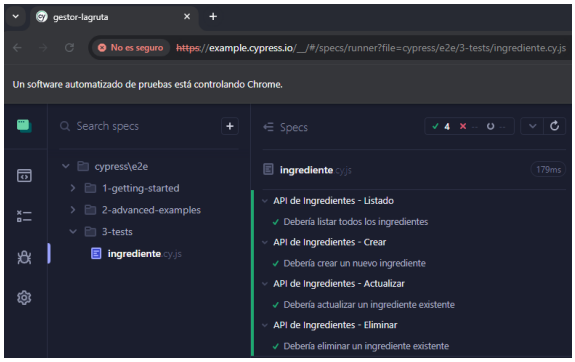

Figura 8.3: Ejecución de las pruebas CRUD sobre la API de Ingredientes en Cypress.

1. **Visibilidad del estado del sistema**: Esta heurística indica que el sistema debe mantener informados a los usuarios en todo momento sobre lo que está sucediendo en la aplicación mediante una retroalimentación apropiada y servida en un tiempo razonable. En la aplicación desarrollada se hace uso de un *ToastContainer* que lanza mensajes emergentes cuando se añade un producto a la comanda o cuando se va a servir o marcar como preparado un plato. Además, que se actualicen las pantallas de Consultar Mesa, Servicio Mesa y Terminal Cocina automáticamente cada vez que hay una modificación en los datos, y sin que el usuario tenga que refrescar la página, asegura que estos puedan ver el estado actualizado de las mesas y de las comandas al instante. Así pues, el sistema cumple con la heurística, ya que los usuarios están constantemente informados sobre el estado del sistema.

2. **Relación entre el sistema y el mundo real**: En esta heurística se evalúa si el sistema habla el lenguaje de los usuarios, es decir, se comunica con él mediante mensajes que utilizan conceptos similares para el usuario, en lugar de términos orientados al sistema. El sistema lanza mensajes de confirmación explicando claramente al usuario la acción que va a aceptar o a rechazar, y los mensajes de éxito también explican al usuario claramente cómo ha cambiado el estado del sistema. La evaluación de esta heurística determina que se cumple, ya que el lenguaje utilizado es claro y conciso.

3. **Control y libertad del usuario**: Esta heurística se basa en la noción de que los usuarios suelen elegir opciones del sistema por error, y que necesitan una salida de emergencia para salir de ese estado rápidamente mediante una opción claramente indicada en la interfaz y sin necesidad de pasar por un diálogo extenso. El usuario puede navegar libremente por la aplicación, si se equivoca durante la navegación tiene disponibles los botones 'Atrás' en varias pantallas para que puedan volver a la pantalla anterior rápidamente. No obstante, en esta versión MVP de la aplicación no se ha añadido una función para hacer retroceder un pedido marcado como servido o como preparado en el caso de que el empleado los haya marcado como error. Una propuesta de mejora en futuras versiones es implementar esa funcionalidad mostrando un histórico de pedidos que le ofrezca la opción al empleado de revertir los cambios.

4. **Coherencia y estándares**: Esta regla indica que los usuarios no tienen que preguntarse si distintas palabras, situaciones o acciones significan lo mismo. La aplicación sigue un orden marcado en la posición de los botones de navegación tanto para acceder a las secciones como para volver 'Atrás'. Además, mantiene una consistencia de colores y de términos utilizados en todas las pantallas, por lo que la interfaz es consistente y cumple con la heurística.

5. **Prevención de errores**: Esta heurística expone que un diseño cuidadoso que prevenga los errores es mucho mejor que programar mensajes de error detallados. La aplicación desarrollada, aún en un estado de versión MVP, se asegura de que al usuario no le salte ningún mensaje de error. La interfaz está programada para que sea casi imposible que se produzcan errores. El usuario tiene una capacidad de modificar el sistema muy limitada, y los formularios de creación y de edición de elementos no permiten la inserción de ningún dato que no coincida con su tipo cuando se crearon las tablas de la BD. Por ese motivo, la aplicación previene los errores antes de que ocurran y cumple con la heurística.

6. **Reconocer en lugar de recordar**: Esta regla sugiere que hay que minimizar la carga de memoria del usuario haciendo visibles los objetos, las acciones y las opciones de la aplicación; es decir, el usuario no tiene por qué recordar información de una parte del diálogo a otra. La aplicación desarrollada muestra listas de mesas, productos y comandas de manera visible, siempre en la misma posición. Además, cada botón de navegación tiene un icono que ilustra la sección a la que se va a acceder cuando se pulse, facilitando el cambio de pantallas con un vistazo rápido. Por ese motivo, cumple con la heurística ya que la interfaz está diseñada para minimizar la carga de memoria del usuario.

7. **Flexibilidad y eficiencia de uso**: Esta heurística indica que la aplicación debe permitir a los usuarios realizar acciones con flexibilidad y eficiencia, ya sea un usuario novato o un usuario experto. La aplicación desarrollada es sencilla, se ha diseñado con una forma simple y directa, por lo que un usuario novato no tendrá problema en utilizarla. No obstante, aunque cumple con la heurística, en versiones futuras se podrían implementar atajos de teclado para usuarios expertos.

8. **Diseño estético y minimalista**: Esta regla es clara, toda interfaz no debe contener información irrelevante o que no se necesite, ya que esa información compite con las unidades relevantes del sistema, disminuyendo así su visibilidad relativa. La interfaz diseñada en esta aplicación es limpia y minimalista, ya que se ha programado para poder utilizarse también desde pequeñas terminales. Por lo tanto, cumple con la heurística ya que su diseño es estético y minimalista.

9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores**: Esta regla indica que los mensajes de error del sistema deben expresarse en un lenguaje claro y sugiriendo al usuario una solución constructiva para salir de ese error. La aplicación ha sido desarrollada para que no muestre ningún mensaje de error y sea fiable en todo momento. Además, debido a las características del entorno de desarrollo de React, en el caso de haber algún problema con la base de datos la aplicación no se bloquea, únicamente lo que sucede es que no se muestran las listas cuando se pretende consultar información de la BD. Por ese motivo, cumple parcialmente la regla, y en futuras versiones sería interesante indicar con algún mensaje de que se ha perdido la conexión con la BD, y que deberían contactar de inmediato con su administrador de sistemas.

10. **Ayuda y documentación**: La última regla indica que, aunque es mejor que un sistema puede utilizarse sin necesidad de documentación, sería útil proporcionar mensajes de ayuda o tutoriales. En esta versión MVP la aplicación no dispone de esas ayudas, por lo que no cumple la regla heurística. En futuras versiones sería interesante incluir alguna sección de ayuda o de documentación para los empleados.

Este análisis nos indica que la aplicación, a pesar de encontrarse en un estado de MVP, cumple con la mayoría de los principios heurísticos de Jakob Nielsen, lo que nos

asegura una experiencia de usuario positiva. No obstante, hay algunos aspectos que se deben tener en cuenta para una versión futura de la aplicación. Aplicar esas mejoras garantizará una mejor usabilidad y eficiencia del sistema.

CAPÍTULO 9
Trabajo futuro

La aplicación desarrollada en este proyecto ofrece una pequeña demostración de todo lo que se puede alcanzar con ella, ya que se ha diseñado con la intención de sacar un mínimo producto viable que pueda enseñarse a clientes y a inversores potenciales. Sin embargo, dista mucho de ser una aplicación completa y totalmente funcional, y tiene un potencial evolutivo muy elevado. Además, este proyecto se ha realizado con pocas nociones de diseño, y podría mejorar mucho con colaboración profesional en ese aspecto.

A continuación se detallan las funcionalidades que se pueden mejorar y las nuevas funcionalidades que se han quedado en el *backlog*.

9.1 Mejora de funcionalidades existentes

- **Registrar Comandas**: La interfaz debe permitir editar la comanda para modificar las cantidades o eliminar alguna línea, y debe implementar un botón de confirmación antes de lanzar los productos a las terminales de los camareros y de cocina. Además, se puede mejorar la interfaz con fotos de los productos para que encontrarlos sea más fácil para el empleado.

- **Servicio a mesas**: La lista de ítems preparados para servir a las mesas podría mostrar la hora en la que el ítem fue añadido a la comanda, para detectar así que hay pedidos que se están retrasando inusualmente y ejecutar una solución de inmediato. Además, podrían agruparse por comanda, para que al camarero le resulte más sencillo marcar en grupo los productos que va a servir a esa mesa en particular.

- **Consultar mesas**: Esta sección debería tener la opción de marcar la mesa como libre una vez se hayan levantado los comensales y hayan pagado la cuenta. Esto archivaría la comanda y permitiría la creación de una nueva asignada a esa mesa.

- **Terminal de cocina**: Se podrían agrupar los platos que hay que cocinar según su tipo para que los cocineros sepan cuántas raciones tienen que preparar de ciertas recetas, lo que mejoraría la eficiencia en cocina. Además, se podría insertar la hora en la que entró el pedido a cocina.

- **Ingredientes**: El panel de ingredientes podría verse beneficiado de una interfaz más refinada y de una tabla más compacta para que sea más fácil hacer ediciones en un dispositivo móvil.

- **Platos caseros**: La interfaz de platos caseros tendría que permitir no solo añadir ingredientes, si no también editarlos. Se podría mejorar mucho la interfaz con un buen diseño.
- **Estadísticas**: En la interfaz de estadísticas se deberían mostrar más cuadros y tablas para informar a la gerencia del restaurante datos como: total de ingredientes que se desperdiciaron por haber llegado a la fecha de caducidad, consumo medio durante los días de cada semana, y otras gráficas de interés para un comercio gastronómico.

9.2 Nuevas funcionalidades

- **Sistema de roles**: Los tres actores de la aplicación deberán tener sus roles y solo podrán acceder a las secciones que tienen asignadas y realizar las acciones que fueron especificadas solo para ellos.
- **Platos caseros**: Faltaría implementar un formulario para poder dar de alta nuevas recetas.
- **Productos comprados**: Implementar un formulario para poder gestionar el inventario de los productos comprados a los proveedores del restaurante.
- **Subir tarifa**: Sería muy útil desplegar un sistema para actualizar cualquier elemento de la base de datos mediante hojas Excel, cuya edición en ordenadores es más sencilla y rápida que en dispositivos móviles. Por ejemplo se podrían actualizar las tarifas del restaurante, el inventario de ingredientes y productos, o modificar y añadir recetas.
- **Sistema de notificaciones**: Sistema de notificaciones para el gerente cuando haya ingredientes cuya fecha de caducidad está próxima, o cuando haya platos que se están retrasando en la preparación o no estén siendo servidos en un marco de tiempo razonable.

CAPÍTULO 10
Conclusiones

Nos encontramos ante un proyecto con mucho potencial y que puede hacer grandes cosas para evitar el desperdicio alimentario en la hostelería. La aplicación permite llevar un control de ingredientes y de ordenarlos por fecha de caducidad, permitiendo así ver de un solo vistazo qué ingredientes gastar antes que otros, o decidir qué hacer con ellos antes de que tengan que ir a la basura.

Además, el uso de tablas estadísticas ofrece un amplio abanico de oportunidades para mostrar qué ingredientes se gastan más y qué platos son los preferidos de los clientes. Tener acceso a estas métricas da paso a otras nuevas, como por ejemplo hacer un estudio de los platos más demandados y determinar qué cantidad se dejan los comensales en el plato para hacer los ajustes necesarios a la hora de medir las raciones. El hecho de que se haya implementado un panel de estadísticas que la gerencia del restaurante pueda consultarlas con un solo clic hace que esta herramienta tenga un potencial inimaginable.

La aplicación desarrollada está planteada como una solución económica y fácilmente personalizable para cualquier tipo de negocio del sector gastronómico. Además, el uso de terminales para mantener comunicados a los camareros y al personal de cocina sin necesidad de gritar las comandas puede catapultar al éxito a este tipo de negocios.

Para cumplir los objetivos principales de este trabajo se han utilizado tecnologías modernas y se han aplicado varios principios de usabilidad. El proyecto se planteó con varios objetivos específicos, y uno era diseñar y desarrollar una interfaz de usuario clara e intuitiva. Para ello se ha diseñado una interfaz que cumple con los principios heurísticos de Jakob Nielsen y que además es fácil de utilizar. Otro objetivo era implementar una API robusta para la gestión de la base de datos, para lo cual se ha creado una API RESTful utilizando Node.js y SQLite para soportar operaciones CRUD sobre distintos elementos de la carta. Esta API ha sido validada mediante pruebas automatizadas con Cypress para garantizar su fiabilidad y su rendimiento. El objetivo de montar un sistema eficiente y que responda rápidamente a las acciones de los usuarios también se ha cumplido mediante la implementación de WebSockets, que proporciona una actualización en tiempo real en todas las interfaces de usuario.

Durante el desarrollo del proyecto se encontraron varios desafíos y problemas, como la integración de WebSockets en la aplicación. Cómo desarrollar un sistema que mostrara la información cambiante en pantalla sin necesidad de refrescarla fue un desafío y requirió de una investigación exhaustiva de y muchas pruebas iterativas hasta lograr una implementación exitosa. Uno de los errores más importantes ha sido subestimar la com-

plejidad a la hora de implementar ciertas funcionalidades, y estos desafíos han supuesto una las mayores curvas de aprendizaje del proyecto.

En lo que respecta a ese aprendizaje, con este proyecto se ha adquirido una cantidad considerable de destrezas y nuevos conceptos. Trabajar con React y Node.js, junto con todo el amplio abanico de herramientas y librerías disponibles como WebSockets, abre la puerta a futuros desarrollos de software con nuevas ideas inspiradas en todo lo que se puede hacer con estas tecnologías.

En conclusión, con este proyecto se ha demostrado la capacidad de integrar conocimientos de varias disciplinas para resolver un problema complejo. Profesionalmente, este trabajo ha ampliado considerablemente mi competencia en el desarrollo de aplicaciones web, las pruebas automatizadas, el diseño de interfaces de usuario y la metodologías de desarrollo ágil. En resumen, todos los objetivos del proyecto se han superado con éxito y se ha desarrollado una solución MVP robusta y eficiente para la gestión del desperdicio alimentario en la hostelería con un amplio margen de mejora en futuros trabajos.

Bibliografía

[Ara23] Valeria Ara. ¿Qué es el ACID?, 2023. Consultado en: https://msmk.university/big-data/que-es-el-acid-msmk-university, Publicado por Valeria Ara el 5 de octubre de 2023.

[Avo24] Avocaty. Vende más y mejor en tu restaurante con Avocaty, 2024. Consultado en: https://avocaty.io/.

[Axi24] Axios. Axios: Promise based HTTP client for the browser and node.js, 2024. Consultado en: https://axios-http.com/es/docs/intro.

[Cha24] Chart.js. Simple yet flexible JavaScript charting for designers & developers, 2024. Consultado en: https://www.chartjs.org/.

[Cyp24] Cypress. JavaScript End to End Testing Framework, 2024. Consultado en: https://www.cypress.io/.

[Dep24] Departamento de Comunicación UEMC Business School. ¿qué es scrum? conoce el framework que agiliza el trabajo en equipo, 2024. Consultado en: https://www.escueladenegociosydireccion.com/revista/business/scrum-framework-agiliza-trabajo-equipo/.

[Dig24] Digital55. ¿Qué son Single Page Application (SPA)? Desarrollo elegido por Gmail y LinkedIn, 2024. Consultado en: https://digital55.com/blog/que-son-single-page-application-spa-desarrollo-elegido-por-gmail-linkedin/.

[dlUE24] Consejo de la Unión Europea. Día internacional contra el desperdicio alimentario: ¿qué hace la ue para evitarlo?, Junio 2024. Consultado en: https://spanish-presidency.consilium.europa.eu/es/noticias/que-hace-ue-reducir-desperdicio-alimentario/.

[Est22] Estado de JavaScript en 2022. State of JS, 2022. Consultado en: https://www.innusual.tech/insights/state-of-js/.

[Eur24] Eurostat. Estimación del desperdicio alimentario y prevención del desperdicio, Junio 2024. Consultado en: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Food_waste_and_food_waste_prevention_-_estimates.

[Exp24] Express. Fast, unopinionated, minimalist web framework for Node.js, 2024. Consultado en: https://expressjs.com/.

[FAO24] FAO. Impacto medioambiental y económico del desperdicio alimentario, Junio 2024. Consultado en: https://www.fao.org/newsroom/detail/FAO-UNEP-agriculture-environment-food-loss-waste-day-2022/es.

[FGE14] K. Fawagreh, M. M. Gaber, and E. Elyan. Random forests: from early developments to recent advancements. *Systems Science & Control Engineering*, 2(1):602–609, 2014.

[Fla24] Flaticon. Iconos de uso gratuito, 2024. Consultado en: https://www.flaticon.com/.

[Har17] PdB Harrington. Automated support vector regression. *Journal of Chemometrics*, 31:e2867, 2017.

[Hop88] J. J. Hopfield. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5):3–10, 1988.

[Ins24] Instituto Nacional de Estadística (INE). Locales por ccaa, actividad principal (grupos cnae 2009) y estrato de asalariados, 2024. Consultado en: https://www.ine.es/jaxiT3/Datos.htm?t=294.

[Jav24a] JavaScript. JavaScript.com: Learn JavaScript and stay connected with the latest news, 2024. Consultado en: https://www.javascript.com/.

[Jav24b] JavaScript.info. WebSocket, 2024. Consultado en: https://es.javascript.info/websocket.

[JSO24] JSON. JavaScript Object Notation (JSON), 2024. Consultado en: https://www.json.org/json-es.html.

[Mar24] Julia Martins. ¿qué son los objetivos smart?, 2024. Consultado en: https://asana.com/es/resources/smart-goals.

[MDN24] MDN Web Docs. Glosario de MDN Web Docs - CRUD, 2024. Consultado en: https://developer.mozilla.org/es/docs/Glossary/CRUD.

[Mic24] Microsoft. Por qué las empresas se benefician del uso de las herramientas de inteligencia empresarial, Junio 2024. Consultado en: https://powerbi.microsoft.com/es-es/what-is-business-intelligence/.

[Nie92] J. Nielsen. Finding usability problems through heuristic evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380, June 1992.

[Nod24] Node.js. JavaScript runtime built on Chrome's V8 JavaScript engine, 2024. Consultado en: https://nodejs.org/en.

[npm24] npmjs. react-toastify, 2024. Consultado en: https://www.npmjs.com/package/react-toastify.

[Pel19] Pello Xavier Altadill Izura. *Desarrollo web con React*. ANAYA MULTIMEDIA, ES, 2019. Encuadernación: Tapa blanda, Idioma: Castellano, Número de páginas: 344, Fecha de lanzamiento: 26/09/2019.

[Per24] Diogo Xavier Ribeiro Pereira. Going zero waste in canteens: Exploring food demand using data analytics, Junio 2024. Consultado en: https://hdl.handle.net/10216/114088.

[Ran24] Randstad. Cuatro de cada diez empresas han visto crecer su rotación laboral en 2022, Junio 2024. Consultado en: https://www.randstad.es/nosotros/sala-prensa/cuatro-de-cada-diez-empresas-han-visto-crecer-su-rotacion-laboral-en-2022/.

[Rea24] React. A JavaScript library for building user interfaces, 2024. Consultado en: https://react.dev/.

[Red23] Red Hat. ¿Qué es una API REST?, 2023. Consultado en: https://www.redhat.com/es/topics/api/what-is-a-rest-api, Actualizado 31 de julio de 2023.

[Red24] Redux. A Predictable State Container for JS Apps, 2024. Consultado en: https://redux.js.org/.

[SL24] B. Sánchez Lang. Reducir el desperdicio alimentario en mercavalencia: indicadores para formular una estrategia, Junio 2024. Consultado en: http://hdl.handle.net/10251/197648.

[SQL24] SQLite. SQLite: Database Software Library, 2024. Consultado en: https://www.sqlite.org/.

[Too24] Too Good To Go. Salva buena comida del desperdicio, 2024. Consultado en: https://www.toogoodtogo.com/es/.

[Uni18] Universitat Politècnica de València - UPV. Diagrama de casos de uso, Noviembre 2018. YouTube https://www.youtube.com/watch?v=orvAkFFWo5o.

[Uni22] Universitat Politècnica de València - UPV. ¿Cómo identificar actores en un diagrama de casos de uso?, Febrero 2022. YouTube https://www.youtube.com/watch?v=d4BS9vMtCWA.

[Uni24a] Naciones Unidas. Objetivo 12: Garantizar modalidades de consumo y producción sostenibles, y sus metas, Junio 2024. Consultado en: https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/.

[Uni24b] Naciones Unidas. Objetivos y metas de desarrollo sostenible de las naciones unidas, Junio 2024. Consultado en: https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/.

[Uni24c] Universidad Europea. Producto Mínimo Viable (PMV), 2024. Consultado en: https://universidadeuropea.com/blog/producto-minimo-viable/.

[Vit24] Vite. Next Generation Frontend Tooling, 2024. Consultado en: https://vitejs.dev/.

APÉNDICE A
Objetivos de Desarrollo Sostenible

Este Trabajo de Fin de Grado está relacionado con los siguientes ODS:

- **ODS 2: Poner fin al hambre**: Al reducir el desperdicio alimentario podemos conseguir contribuir un poco más a poner fin al hambre. Esta aplicación está desarrollada para ofrecer una herramienta a los servicios de restauración que también tienen que poner de su parte y colaborar.
- **ODS 8: Trabajo decente y crecimiento económico**: El desarrollo de una aplicación para la gestión eficiente de comandas y mesas en un restaurante puede mejorar significativamente la productividad y eficiencia del personal. Al facilitar la gestión y el seguimiento de las comandas se puede reducir el tiempo de espera y aumentar la satisfacción del cliente, lo que podría suponer un aumento del volumen de negocio y el crecimiento económico. Además, la digitalización de estos procesos puede generar nuevas oportunidades de empleo en el sector tecnológico para el soporte y mantenimiento de la aplicación.
- **ODS 9: Industria, innovación e infraestructura**: La implementación de una solución tecnológica avanzada como esta puede ser vista como una innovación en la industria de la restauración. Facilita la adopción de tecnologías modernas como WebSockets para actualizaciones en tiempo real y una interfaz de usuario intuitiva desarrollada con React, lo que promueve la construcción de infraestructuras sólidas y sostenibles. Gracias a esto se fomentará la innovación y mejorará la eficiencia operativa del negocio.
- **ODS 12: Producción y consumo responsables**: La digitalización de los procesos de gestión de comandas y mesas puede reducir significativamente el uso de papel y otros recursos físicos, promoviendo prácticas sostenibles. Al minimizar el desperdicio de recursos materiales y optimizar la gestión de inventarios, la aplicación contribuye a una producción y consumo más responsables dentro del restaurante.
- **ODS 4: Educación de calidad**: El desarrollo de esta aplicación ha implicado un proceso de aprendizaje significativo en el uso de tecnologías modernas y metodologías ágiles. Este conocimiento adquirido puede ser compartido y utilizado en entornos educativos para formar a futuros desarrolladores. Además, este proyecto puede servir como un caso de estudio para otros estudiantes de informática y de desarrollo de software.
- **ODS 10: Reducción de las desigualdades**: La aplicación facilita la gestión eficiente de un restaurante, lo que puede beneficiar a negocios pequeños y medianos que no tienen acceso a soluciones tecnológicas que les resulten caras. Al proporcionar una

herramienta asequible y efectiva, se contribuye a reducir las desigualdades en el acceso a tecnología avanzada en el sector de la restauración.

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**		X		
ODS 3. **Salud y bienestar.**			X	
ODS 4. **Educación de calidad.**		X		
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**		X		
ODS 11. **Ciudades y comunidades sostenibles.**				
ODS 12. **Producción y consumo responsables.**	X			
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Figura A.1: Relación del TFG con los Objetivos de Desarrollo Sostenible.

APÉNDICE B
Glosario

- Aplicación web: Programa informático alojado en un servidor que se ejecuta en un navegador a diferencia de las aplicaciones de escritorio que deben instalarse localmente en el dispositivo del usuario.

- App: Abreviatura de *application*, término en inglés que se refiere a una aplicación de software diseñada para realizar una función o un conjunto de funciones específicas.

- Backend: Parte del desarrollo de una aplicación oculta al usuario que se encarga de la lógica de negocio, la gestión de la base de datos y la creación de las API que serán utilizadas por el *frontend*.

- Backlog: Lista priorizada de tareas pendientes de realizar en un proyecto de desarrollo ágil. Incluye características, mejoras, correcciones de errores y cualquier otro trabajo necesario. El *backlog* se actualiza y reordena continuamente según las prioridades del proyecto.

- Broadcast: Técnica de comunicación en redes y aplicaciones que envía un mensaje o paquete de datos a todos los dispositivos en un dominio específico de la red o a todos los componentes de una aplicación. En el contexto de desarrollo de software, se puede referir al envío de actualizaciones o notificaciones a todos los clientes conectados, asegurando que todos reciban la información simultáneamente.

- Business Intelligence (BI): Conjunto de estrategias, procesos, aplicaciones y tecnologías que las empresas utilizan para recopilar, integrar, analizar y representar información estadística.

- Feedback: Información o reacciones que se proporcionan como respuesta a un proceso, producto o actuación. En el desarrollo de software, el feedback puede provenir de usuarios finales, clientes, o miembros del equipo de desarrollo, y se utiliza para mejorar la calidad del producto y ajustar las características según las necesidades y expectativas.

- Framework: Conjunto de herramientas y bibliotecas que proporciona una estructura estándar para el desarrollo de software. Facilita y acelera la creación de aplicaciones al ofrecer componentes reutilizables, soluciones predefinidas para problemas comunes y una arquitectura consistente.

- Frontend: Parte de una aplicación que interactúa directamente con el usuario final. Se encarga de la interfaz de usuario y la experiencia del usuario, generalmente desarrollado usando tecnologías como HTML, CSS y JavaScript.

- Machine learning: Disciplina de la inteligencia artificial en la que una máquina es entrenada mediante algoritmos que reconocen patrones que las ayudan a aprender y a mejorar su rendimiento a la hora de hacer predicciones según la entrada de datos.
- Middleware: Componentes de software que actúan como intermediarios en la gestión de las solicitudes entre el cliente y el servidor en una aplicación. Los *middlewares* pueden realizar una variedad de tareas como la autenticación, la autorización, el manejo de errores, el registro de actividades y la manipulación de datos antes de que la solicitud llegue al controlador final.
- Mockups: Representaciones visuales de la interfaz de usuario de una aplicación que muestran el diseño y la funcionalidad esperada. Los *mockups* son más detallados que los *wireframes* y a menudo incluyen colores, tipografía, y elementos gráficos, proporcionando una vista previa precisa de cómo se verá y funcionará el producto final.
- Prototipado: Proceso de creación de versiones preliminares de un producto o sistema que permiten explorar ideas, funcionalidades y diseños antes de desarrollar la versión final. El prototipado puede incluir *mockups*, *wireframes* y prototipos interactivos, y es utilizado para validar conceptos, obtener retroalimentación de los usuarios y realizar ajustes iterativos.
- Renderizar: Proceso de generar una imagen o representación visual a partir de un modelo o datos, generalmente en el contexto de gráficos por computadora o desarrollo web. En el desarrollo web, renderizar se refiere a la creación de la interfaz de usuario visible en el navegador a partir de código HTML, CSS y JavaScript.
- Script: Archivo de texto que contiene una serie de instrucciones o comandos que se ejecutan de forma secuencial para automatizar tareas en un entorno de software.
- Software: Conjunto de programas, datos e instrucciones que permiten a una computadora realizar tareas específicas.
- Sockets: Interfaces de programación que permiten la comunicación entre dos nodos en una red. Los sockets pueden ser utilizados para enviar y recibir datos entre un cliente y un servidor a través de una red, facilitando la transmisión de datos en tiempo real.
- Sprint: Período de tiempo corto durante el cual un equipo de desarrollo de software trabaja intensamente para completar un conjunto de tareas planificadas previamente.
- Stakeholders: Personas, grupos u organizaciones que tienen un interés o están afectados por el resultado de un proyecto. Los *stakeholders* pueden incluir clientes, usuarios finales, gerentes de proyecto, miembros del equipo de desarrollo, patrocinadores y cualquier otra parte interesada en el éxito del proyecto.