



Proyecto de fin de ciclo de desarrollo de aplicaciones multiplataforma curso 2019-2020

Memoria de la aplicación *Machala's Grill* para el control de
comandas en restauración

ALUMNO: Florin Pentelescu

RESUMEN

Durante el estudio del ciclo de desarrollo de aplicaciones multiplataforma hemos aprendido la importancia de crear aplicaciones informáticas que puedan ser utilizadas desde cualquier dispositivo electrónico y que sirvan como herramientas de trabajo que faciliten y mejoren la calidad del trabajo.

Por ello, se han aplicado los conocimientos adquiridos hacia el sector hostelero, centrándose tanto en la mejora de la comunicación de las comandas entre camareros y cocineros, como en la mejora del control del *stock* y de los tiempos de preparación.

Machala's Grill es una aplicación que se adapta perfectamente a cualquier dispositivo electrónico y que contempla el mantenimiento de las tablas de bases de datos que utiliza para su correcto funcionamiento, el control de acceso y el control de las comandas y del *stock*.

Entre las funciones más importantes de la aplicación, destaca la capacidad de actualizar la información en cocina al instante, reduciendo el tiempo de respuesta en comparación con el sistema tradicional de comandas (papel y bolígrafo). Así mismo, permite al cocinero controlar los tiempos de cocción, el tiempo de preparación y el *stock* actualizado de los productos.

En cuanto a la sección gráfica de la aplicación, se han utilizado las siguientes tecnologías:

- Aplicación móvil: AndroidStudio.
- Aplicación de escritorio: NetBeans IDE 8.2.
- Conexión entre la base de datos y las aplicaciones: SQL Server, WebServices.

La programación se ha desarrollado en Java con el entorno de desarrollo integrado NetBeans IDE 8.2 y MariaDB, un sistema de gestión de bases de datos con licencia GLP (General Public License) derivado de MySQL y que forma parte de XAMPP.

ABSTRACT

During the study of the multi-platform application development course, we have learned the importance of creating computer applications that can be used from any electronic device and that serve as work tools that facilitate and improve the quality of work.

For this reason, we have applied the knowledge acquired to the hotel and catering sector, focusing both on improving the communication of commands between waiters and cooks, and on improving the control of stock and preparation times.

Machala's Grill is an application that fits perfectly to any electronic device and includes the maintenance of the database tables it uses for its correct operation, access control and control of commands and stock.

One of the most important functions of the application is the ability to update the information in the kitchen instantly, reducing the response time compared to the traditional system of commands (paper and pen). It also allows the chef to control cooking times, preparation time and updated stock of products.

As for the graphic section of the application, the following technologies have been used:

- Mobile application: AndroidStudio.
- Desktop application: NetBeans IDE 8.2.
- Connection between the database and the applications: SQL Server, WebServices.

The programming has been developed in Java with the integrated development environment NetBeans IDE 8.2 and MariaDB, a database management system with GLP (General Public License) derived from MySQL and part of XAMPP.

ÍNDICE

RESUMEN.....	1
ABSTRACT.....	2
ÍNDICE	3
ÍNDICE DE TABLAS	4
ÍNDICE DE IMÁGENES	5
INTRODUCCIÓN	6
OBJETIVOS.....	8
ESTADO ACTUAL DEL PROBLEMA.....	9
SOLUCIONES PROPUESTAS	10
1. Análisis de requisitos.....	10
2. Tecnologías que se usan	12
3. Modelos.....	13
4. Implementación	14
5. Experimentación.....	14
6. Requisitos de funcionamiento.....	15
CONCLUSIONES	17
TRABAJO FUTURO.....	18
VISIÓN COMERCIAL	19
BIBLIOGRAFÍA.....	23
ANEXOS.....	24
Manual de instalación.....	24
Manual de uso.....	25
Solución de posibles problemas	36

ÍNDICE DE TABLAS

Tabla 1: Ejemplo de historial de comandos.

Tabla 2: Posibles problemas y soluciones.

ÍNDICE DE IMÁGENES

Figura 1: Distribución de la producción en hostelería.

Figura 2: Modelo en cascada.

Figura 3: Esquema general del análisis DAFO.

Figura 4: Análisis DAFO para la aplicación Machala's Grill.

Figura 5: Elementos del marketing mix.

Figura 6: Pantalla de autenticación de la aplicación Android.

Figura 7: Pantalla de inicio de la aplicación de escritorio.

Figura 8: Pantalla de selección de mesa.

Figura 9: Pantalla de selección y personalización de platos.

Figura 10: Funciones de los botones de la pantalla de selección y personalización de platos.

Figura 11: Ventana Dialog para elegir salsa y/o guarnición.

Figura 12: Pantalla de inicio de la aplicación de escritorio

Figura 13: Pantalla de historial de órdenes.

Figura 14: Pantalla de visualización del stock actual.

INTRODUCCIÓN

El sector hostelero es una de las principales fuentes de ingresos de España ya que factura anualmente más de 100.000 millones de euros y aporta aproximadamente el 6% del PIB anual. Según las estimaciones del Instituto Nacional de Estadística en el año 2019, de esta importante suma de dinero, aproximadamente el 68% de los ingresos proceden de restaurantes y bares.

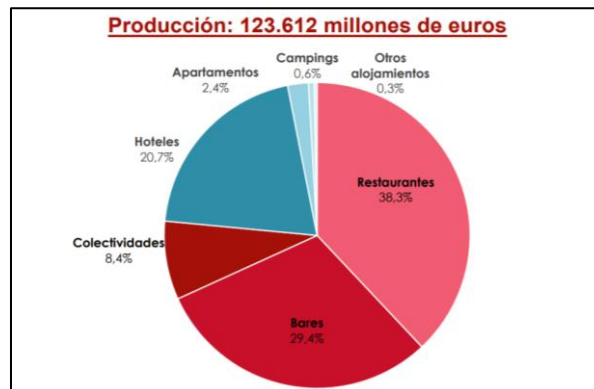


Figura 1. Distribución de la producción en hostelería

Por otro lado, es importante recalcar que la restauración ofrece trabajo a más de 1.7 millones de trabajadores (principalmente durante los meses de verano) cada año. Es bien sabido que las personas que trabajan en este sector suele encontrarse sometidas a largas jornadas de trabajo bajo condiciones estresantes. Tras analizar estos datos y teniendo en cuenta la cantidad de personas con riesgo a sufrir estrés que se dedican a trabajar en este sector surgió la idea de desarrollar una aplicación que facilite el trabajo diario de camareros y cocineros.

Al utilizar *Machala's Grill app*, parte de la carga emocional de los trabajadores podría verse reducida ya que se optimizaría el tiempo de trabajo de los camareros haciendo que se comuniquen instantáneamente con cocina sin necesidad de desplazamientos.

Los cocineros también notarían mejoras en su trabajo porque podrían realizar el seguimiento de las comandas de cada mesa, verían el tiempo que llevan con cada una de ellas y podrían tener un control del *stock* real y actualizado. Por otro lado, se verían reducidos los errores de comandas derivados del sistema de bolígrafo tradicional ya que en ocasiones los camareros no anotan claramente la comanda pudiendo llevar a error de lectura en cocina.

Además, también se mejoraría la calidad de servicio de cara a los comensales debido a que cada mesa llevaría su propio seguimiento pudiendo priorizar el trabajo según el tiempo de espera de cada mesa.

Tras el análisis de la situación actual, la identificación de los puntos débiles del sistema tradicional y el estudio de las necesidades reales de los trabajadores, se procedió a planificar el diseño de la aplicación.

Tras la elaboración de la base de datos se estudió el diseño de las acciones que debería desarrollar la aplicación para ser verdaderamente útil. Se buscó que reflejara coherentemente las tareas a realizar, que cumpliera con los objetivos propuestos y, sobretodo, que respondiera a la problemática y necesidades planteadas desde la realidad de los trabajadores.

Otra fase fundamental para la construcción de la solución definitiva y para la implementación final de la aplicación fue la programación de los módulos *software*.

Tras la realización de todo lo nombrado anteriormente, se procedió a realizar pruebas de funcionamiento de la aplicación, buscando posibles errores y subsanando los problemas que pudiera dar.

OBJETIVOS

El objetivo principal de este proyecto es desarrollar una herramienta informática para la gestión de la realización y preparación de comandas en restauración, que garantice el alcance funcional buscado, la correcta petición y preparación de las comandas y la calidad de servicio buscada en estos establecimientos.

Los objetivos secundarios de la aplicación a desarrollar son los siguientes:

1. Organizar las comandas por orden de entrada.
2. Facilitar la lectura de la comanda en cocina.
3. Reducir los costes económicos derivados de errores humanos.
4. Mejorar la atención al cliente reduciendo el tiempo de preparación de las comandas.
5. Optimizar la gestión interna de comandas.

Así mismo, durante la realización del proyecto, se establecieron tres objetivos más que se consideran de gran importancia:

1. Agilizar el trabajo de los trabajadores de hostelería para reducir su carga de estrés laboral.
2. Realizar un adecuado control de *stocks*.
3. Sintetizar todos los conceptos adquiridos durante el grado demostrando su utilidad en cualquier ámbito profesional.

ESTADO ACTUAL DEL PROBLEMA

El problema actual deriva de la necesidad que tienen los establecimientos de restauración de procesar adecuadamente la información de los clientes y su correcta transmisión a cocina.

Desde un punto de vista práctico, el correcto registro, transmisión a cocina y control de preparación de las comandas supondría una mejora importante en la calidad de servicio de un restaurante.

Cada vez más, estos establecimientos apuestan por la informática para mejorar su gestión interna y así sustituir el método tradicional, que va quedando obsoleto y poco eficaz frente a los nuevos recursos informáticos.

Es evidente que el procedimiento actual basado en el bolígrafo y papel puede llevar a cometer numerosos errores humanos, como errores de lectura, pérdida de las comandas o desorden de los papeles. Además, la falta de comunicación actual entre camareros y cocina derivada, mayoritariamente, de la falta de tiempo en el trabajo, lleva a cometer errores de cocinado porque información valiosa para el cocinero se puede perder por el camino.

Otro de los principales problemas actuales, es la dificultad de los cocineros para memorizar las comandas al mismo tiempo que controlan los tiempos de cocción de cada plato y el stock del restaurante.

Esta aplicación no solo supondría mejoras en el tiempo de transmisión de información entre camareros y cocina, además, ayudaría a los cocineros a llevar un mejor control de los platos que están preparando ya que podrán ver qué mesa tiene prioridad y el tiempo que han dedicado a cada preparación. Por otro lado, se podría llevar un registro de stocks actualizado en tiempo real.

SOLUCIONES PROPUESTAS

La solución propuesta para los problemas detectados contempla en su totalidad tanto el objetivo principal como los objetivos secundarios.

Durante el desarrollo del proyecto, se ha buscado la creación de una aplicación fácil de utilizar, que disponga de pantallas claras y de fácil comprensión para que el personal de los restaurantes pueda utilizar la aplicación sin dificultad. También se ha tenido en cuenta el tipo de rutina de los trabajadores por lo que se ha planteado el proyecto para que los camareros puedan utilizar la aplicación desde un teléfono móvil inteligente o una tableta que puedan llevar encima mientras que los cocineros la utilicen desde unas pantallas fijas en cocina.

Se presenta un desarrollo completamente gráfico y fácil de utilizar. Se ha potenciado el uso de controles tipo JTable, JButton, JFrame, JPanel... Que reducen al mínimo la necesidad de utilizar teclado y facilitan la utilización del programa en los dispositivos táctiles inteligentes.

Tanto la aplicación de escritorio como la aplicación móvil, permiten mantener la totalidad de las tablas necesarias para su correcto funcionamiento y proporcionan un control de acceso seguro basado en usuarios y contraseñas.

1. Análisis de requisitos

Machala's Grill app ofrece la posibilidad de tomar comandas de forma rápida y sencilla, mediante un móvil o una tableta, enviarlas directamente al receptor, en este caso una pantalla táctil, en ella el cocinero puede ver los pedidos recibidos, el tiempo que lleva abierto desde el momento que se tomó en la mesa, el camarero responsable en caso de error, además de temporizadores que avisan al cocinero de la duración de la cocción de un plato.

Al realizar todos los platos pedidos en una mesa, la comanda queda archivada en un historial que permite realizar comprobaciones en caso de necesidad.

Es importante recordar que para conseguir los objetivos buscados con esta aplicación, se debe tener en cuenta que los usuarios deben tener los estudios en cocina que se requieren para sacar adelante un correcto servicio al cliente.

Se han dividido los requisitos a tener en cuenta en requisitos del usuario, requisitos del establecimiento y requisitos del sistema.

Requisitos del usuario

Se desarrollará una aplicación intuitiva y de fácil uso que no requiera una formación técnica exhaustiva del personal que vaya a utilizarla. Pese a ello, será imprescindible que el personal conozca la aplicación y tenga suficiente destreza para utilizarla correctamente y sacar el máximo provecho de ella.

Por otro lado, se requiere que el personal esté suficientemente formado en su área para poder utilizar la información correctamente (es una realidad que en numerosos establecimientos españoles el personal de restauración no tenga los estudios necesarios para ejercer su profesión). La formación en restauración será imprescindible para poder utilizar correctamente el lenguaje técnico.

Requisitos del establecimiento

La aplicación se adaptará a las necesidades de los establecimientos de restauración, de manera que estos no deberían cambiar demasiado la manera de trabajar y de organizarse.

Los establecimientos deberán implantar la aplicación en soportes físicos que proporcionaran a sus empleados. Deberán ofrecer, al menos, un dispositivo móvil con sistema Android (Smartphone o Tablet) y pantalla en cocina con sistema operativo Windows. También será imprescindible una buena conexión a internet para que la aplicación funcione correctamente.

Hay que recalcar que la entrada de datos puede realizarse a través de varios dispositivos.

Requisitos del sistema

El software deberá ser capaz de mandar comandas y tener las existencias actualizadas en tiempo real, además de tener una interfaz gráfica con la que se pueda interactuar y ofrecer apoyo al personal.

Los requisitos de software y hardware serán identificados claramente y serán comunicados al cliente. Los requisitos del sistema se ajustarán todo lo posible a la realidad tecnológica. Pese a ello, se exigirán unos requisitos mínimos para garantizar que el programa informático funcione y rinda adecuadamente.

Para que la aplicación sea funcional se necesitará:

1. Un dispositivo móvil para la toma de comanda a través de una aplicación Android (versión Android 11).

2. Un ordenador para la instalación de la aplicación de escritorio con los siguientes requisitos mínimos:

- Procesador: 4 núcleos y 2.4 GHz.
- RAM: 4 GB para 32 bits o 8 GB para 64 bits.
- Espacio en disco duro: 16 GB para un SO de 32 bits o 32 GB para un SO de 64 bits.
- Tarjeta gráfica: DirectX 11
- Pantalla: Full HD.
- Conexión a Internet.
- Sistema operativo: Windows 10.

3. Un servidor en Apache Server.

2. Tecnologías que se usan

Para el desarrollo de la aplicación se han utilizado tecnologías estándar y herramientas de código abierto. Para la base de datos, se ha utilizado el gestor de bases de datos gratuito MariaDB.

Toda la parte de programación de la aplicación se ha realizado mediante **Java**.

El desarrollo del software se ha realizado en un ordenador que dispone de: Microsoft Windows 10, 64 bits, procesador Intel i7 con memoria RAM de 32 GB, disco duro SSD de 2TB y monitor con resolución full HD.

Como entorno de desarrollo integrado, se ha elegido NetBean IDE 8.2.

El software para la aplicación móvil se ha desarrollado con la herramienta **AndroidStudio**, disponible tanto para Windows, MacOS como ChromeOS y Linux.

Android Studio permite realizar numerosas acciones relacionadas con las aplicaciones para Android como, por ejemplo, analizar aplicaciones similares, ejecutar aplicaciones, crear nuevas aplicaciones y ser utilizado como emulador de Android. En cuanto a su apartado de creación de aplicaciones, ofrece herramientas y servicios imprescindibles para la creación de nuevas aplicaciones para Android, además del código y diseño de la interfaz de usuario de la aplicación.

Por último, se ha utilizado **WebServices**, que consiste en un conjunto de protocolos y estándares que sirven para intercambiar datos entre distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma.

3. Modelos

Para garantizar que el resultado final cumpla con los objetivos planteados, es necesario identificar adecuadamente las entidades a partir de la realidad a informatizar así como los requerimientos del usuario de la aplicación.

Una vez identificadas las realidades a mejorar, se debe diseñar adecuadamente el proyecto para a la hora de construir las tablas todo encaje y así la implementación del proyecto sea la correcta.

Para desarrollar el proyecto siguiendo se ha utilizado es el modelo en cascada, ya que es un método de desarrollo lineal y secuencial que garantiza el cumplimiento de las distintas fases para lograr el cumplimiento de los objetivos propuestos.

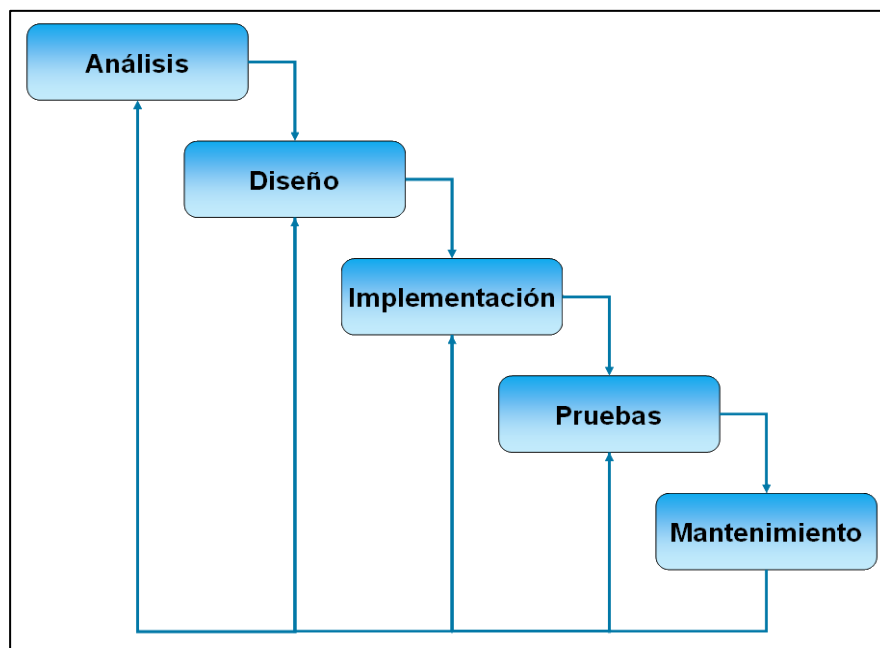


Figura 2. Modelo en cascada

4. Implementación

Para el desarrollo de la aplicación se han utilizado las tecnologías explicadas previamente en el apartado 10.2 (Tecnologías que se usan).

Se han incorporado al entorno NetBeans IDE 8.2 las siguientes bibliotecas:

- Mysql conector java 5.1 22 bin jar.
- Javacsv.jar.

El procedimiento general establecido para la implementación de la aplicación ha sido el siguiente:

1. Análisis de la infraestructura para la posterior puesta en marcha. Para ello ha sido necesario determinar los recursos tecnológicos y el software necesarios para la instalación de la aplicación.
2. Evaluación de los recursos disponibles para la puesta en marcha.
3. Garantizar que los recursos disponibles en el nuevo entorno son compatibles con la aplicación.
4. Configuración de la base de datos instalando las entidades necesarias para un correcto funcionamiento.
5. Definición de los distintos usuarios dentro de la aplicación y determinación de las contraseñas de acceso.
6. Personalización de la configuración de la aplicación para que cumpla los requisitos de los usuarios.
7. Realización de pruebas que garanticen el correcto funcionamiento.
8. Puesta en funcionamiento del software.
9. Evaluación de los resultados y documentación de los errores.
10. Mantenimiento y realización de actualizaciones.

5. Experimentación

La realización del proyecto se ha realizado siguiendo distintas fases:

- Conocimiento de la realidad: se ha buscado documentación y datos relacionados con el proyecto. Además de entrevistas con trabajadores y evaluación de experiencias personales.
- Diseño de la base de datos que refleje las necesidades identificadas en el análisis de la realidad, así como sus relaciones y atributos, que se correspondan con el modelo de datos y dependencias.
- Diseño de los procesos que deben estar disponibles en la aplicación para hacerla funcionar correctamente y que refleje de manera coherente las distintas

tareas necesarias para cumplir con los objetivos propuestos, respondiendo a la problemática y necesidades reales.

- Programación del software necesario para la implementación final del proyecto y para la construcción de la solución informática.
- Realización de pruebas que verifiquen el correcto funcionamiento de la aplicación.
- Control de la calidad de la aplicación.
- Redacción de la memoria final.

Previo al desarrollo del proyecto, se realizaron pruebas de controles y de componentes Java (JTable, JFrame, JButton...) como ejercicio personal.

También se construyeron módulos para conectar la base de datos y así poder probar el acceso desde NetBeans y AndroidStudio para ver el comportamiento de las sentencias SQL más comunes (INSERT, UPDATE, DELETE...)

Los principales problemas que han surgido durante el desarrollo del proyecto son los siguientes:

- Creación de paneles individuales por número de mesa en tiempo real.
- Correcta actualización de GUI (Graphical User Interface) tras la importación de nuevos datos.
- Adaptación de los Java Swing Layouts a la aplicación.

Se subsanaron tras numerosas horas de trabajo intensivo.

6. Requisitos de funcionamiento

El camarero realiza la toma de la comanda y la envía a través de un dispositivo móvil Android. Este dispositivo contiene la información de los productos disponibles actualizada por los responsables de stock.

La aplicación guarda la información en una base de datos SQL para que el programa de escritorio en Windows pueda capturar los datos guardados y reflejarlos en la pantalla de cocina.

En dichas pantallas, quedará impresa la comanda con toda la información que haya introducido el camarero (número de mesa, persona que atiende, platos a elaborar, cocción de las carnes, guarniciones...)

Las funciones del software móvil en la entrada de datos se estructuran por fases:

1. Para iniciar la aplicación el empleado debe autenticar su identidad.
2. A continuación deberá seleccionar el número de la mesa que esté atendiendo.
3. Por último, podrá personalizar los platos al gusto del cliente y registrar el pedido en la base de datos.

En la aplicación de escritorio, una vez realizada la autenticación, se selecciona el acceso a comandas o a la administración.

Tras seleccionar la opción de comandas se genera en el menú inicial una tercera sección donde se puede visualizar el historial diario de comandas. Al ejecutar el panel de comandas en vivo, se realiza automáticamente un intento de recuperación de comandas en caso de haber habido un fallo inesperado como un corte de luz, caída de la red.. Si el programa detecta datos en la tabla de mesas incompletas, las recupera y las imprime en pantalla para continuar con el trabajo y queda a la espera de nuevas órdenes. Existe la función de “cerrar día” que genera un archivo CSV con el historial de mesas completadas.

Dentro del apartado de administración se encuentra la visualización del stock disponible, la posibilidad de añadir otro pedido y la opción de edición de los platos de la carta.

CONCLUSIONES

La creación del proyecto ha supuesto una experiencia muy interesante. Se ha trabajado en profundidad y por completo con la programación de una aplicación que responde a una necesidad real. Esto ha significado una unificación de los conceptos estudiados durante el ciclo formativo que ha servido para entender completamente el desarrollo de una aplicación multiplataforma.

La experiencia ha sido sumamente gratificante ya que se ha entendido que la informática puede aportar a las empresas nuevos valores para mejorar su organización y funcionamiento internos.

Concluyendo, se puede confirmar que se han cumplido los objetivos propuestos para la aplicación.

TRABAJOS FUTUROS

La realización de un proyecto de esta magnitud supone que siempre sea posible realizar mejoras en la aplicación. Por ello se proponen los siguientes trabajos futuros:

- Posibilidad de deshacer/rectificar una comanda: Sería muy interesante desarrollar la opción de deshacer o rectificar una comanda y que se actualice instantáneamente en cocina.
- Aviso de fin de preparado: Se podría añadir la opción de avisar a los camareros cuando la comanda esté lista para salir de cocina, de manera que el camarero supiera que ya se ha finalizado el preparado.
- Aviso de alergias/intolerancias: Cada día es más importante realizar una correcta gestión de las alergias e intolerancias de los consumidores. Por ello se propone avisar a cocina de que en alguna mesa hay un alérgico o intolerante a algún alimento.
- Creación de una base de datos completa con el control total del stock.
- Hacer funcional el botón de editar carta dónde el encargado pueda eliminar o añadir nuevos platos a la carta.
- Desde cocina poder invocar una comanda completada.

VISIÓN COMERCIAL

Machala's Grill es una aplicación que mejora la gestión de las comandas en un establecimiento de restauración. Se puede utilizar en restaurantes con servicio a la carta. Se considera que utilizando esta aplicación se puede realizar un mejor control y gestión del servicio y negocio.

Se parte de la idea de digitalizar las comandas, ofrecer apoyo al cocinero a la hora de gestionar los puntos de cocción de las elaboraciones y realizar una toma de comandas sencilla, rápida y fiable.

En cuanto a la visión de futuro, se desea implantar un sistema informático que brinde un control completo sobre el servicio del restaurante, la gestión del stock, los horarios del personal incluso gestione las cámaras de seguridad del local y el registro de ingresos diarios.

Un análisis DAFO es un estudio completo que analiza las debilidades, amenazas, fortalezas y oportunidades de un proyecto o negocio. Este tipo de análisis puede ser aplicado en cualquier tipo de proyecto independientemente de su tamaño, actividad o área de negocio y es una herramienta necesaria cuando se plantea un proyecto empresarial ya que puede ser de utilidad en el momento de establecer estrategias de actuación.



Figura 3. Esquema general del análisis DAFO

Un buen análisis DAFO se divide en dos secciones principales:

1. Análisis interno de las fortalezas y debilidades que presenta el proyecto.
2. Análisis externo de las amenazas y oportunidades que ofrece el mundo exterior y que deben ser tenidas en cuenta para superarlas o aprovecharlas.

Se ha realizado el estudio DAFO para la aplicación obteniendo los siguientes resultados:





































































 Debilidades 	 Amenazas 
<ul style="list-style-type: none">  Empresas con pocos recursos económicos    Elevada carga de trabajo de los trabajadores    Dependencia de buena conectividad    Posibles problemas de compatibilidad    Los restaurantes no poseen los dispositivos requeridos   <p>Añadir debilidad +</p>	<ul style="list-style-type: none">  Empresas con elevada competencia    Posible falta de entusiasmo del personal    Posible desconfianza del sector hacia la digitalización   <p>Añadir amenaza +</p>
 Fortalezas 	 Oportunidades 
<ul style="list-style-type: none">  Personal dispuesto a aceptar mejoras    Uso sencillo e intuitivo    Transmisión de comandos instantánea    Inexistente confusión de lectura    Gran ayuda al cocinero (cocciones, tiempos, indicadores...)    Mayor control del administrador    Gran capacidad de adaptación al usuario    Atención personalizada al cliente    Software Open Source que no requiere un coste adicional para el cliente   <p>Añadir fortaleza +</p>	<ul style="list-style-type: none">  Mejora de gestión rápida    Baja inversión    Gran demanda de digitalización del sector   <p>Añadir oportunidad +</p>

Figura 4. Análisis DAFO para la aplicación *Machala's Grill*

Tras la realización del estudio DAFO para la aplicación *Machala's Grill*, se observa que la aplicación tiene más fortalezas y oportunidades que debilidades y amenazas por lo que se puede concluir que es un negocio rentable.

En cuanto a las fuentes de financiación, el desarrollo de la aplicación no requiere de financiación. Por otro lado, los establecimientos de hostelería asumirán los costes derivados de la compra de dispositivos electrónicos.

La nueva realidad digital ha llevado a la generación de 4 P's que responden al comportamiento de los usuarios del mundo digital.

Estas 4 P's son: Producto, Precio, *Place* y Promoción.



Figura 5. Elementos del marketing mix

- **Producto:** Explica la necesidad del consumidor y los productos que se necesitan desarrollar para mejorar y añadir valor al producto.
- **Precio:** Esta variable ayuda a posicionar la empresa. Las estrategias de costes justifican el precio estudiado y muestran el margen de beneficio.
- **Place (distribución):** Estudia las fases que atraviesa el producto desde que se crea hasta que llega al consumidor.
- **Promotion (promoción):** Explica la actividad que llevará a cabo la empresa para que el producto llegue al máximo de consumidores. Engloba las acciones publicitarias.

A continuación se procede a aplicar el *Marketing mix* a la aplicación propuesta.

Producto: la aplicación sirve para mejorar la gestión de las comandas en un restaurante así como la preparación de los platos y el control del stock. Está diseñada de manera fácil e intuitiva y pretende cubrir las necesidades generales de cualquier establecimiento hostelero. Junto con la compra de la aplicación se ofrece un servicio de mantenimiento de la misma para añadir nuevas funciones y subsanar posibles errores.

Precio: el precio de adquisición será de 100 €.

Distribución: la aplicación está pensada para ser utilizada en establecimientos hosteleros del territorio español. La adquisición de la aplicación no requiere de intermediarios.

Promoción: La promoción de la aplicación se basará en realizar un video explicativo promocionando la aplicación y enviándolo por correo electrónico a los negocios. También se irá personalmente para presentar el producto.

BIBLIOGRAFÍA

www.hosteltur.com/economía

Anuario de la hostelería en España 2019

<https://dafo.ipyme.org>

<https://economipedia.com>

<http://www.lawebdelprogramador.com>

<https://stackoverflow.com>

<https://docs.oracle.com>

<https://www.php.net>

<https://desarrolloweb.com>

<https://www.javatpoint.com>

<https://www.w3schools.com>

ANEXOS

Manual de instalación

El procedimiento a seguir para una correcta instalación de la aplicación es el siguiente:

1. Instalar en el ordenador el entorno de desarrollo integrado NetBeans IDE 8.2.
2. Instalar en el ordenador AndroidStudio.
3. Instalar el servidor XAMPP en el equipo.
4. Descomprimir el paquete “proyecto.rar” que contendrá la carpeta del proyecto NetBeans, la carpeta del proyecto AndroidStudio, la base de datos y la carpeta de consultas php.
5. Importar en NetBeans IDE 8.2 la biblioteca conector-java-5.1.22-bin.jar.
6. Importar en NetBeans IDE 8.2 la biblioteca javacsv.jar.
7. Abrir la aplicación Android en AndroidStudio y actualizar en todas las funciones que contengan conexión a la base de datos la IP del ordenador en el que se ha instalado XAMPP.
8. En la ubicación de instalación de XAMPP, copiar en la carpeta HTDOCS la carpeta de consultas php.
9. Importar la base de datos de la aplicación partiendo del fichero “ordenes.sql”.
10. Crear un acceso directo en el escritorio del ejecutable del ProyectoEscritorio.

Tras seguir estos pasos, la aplicación puede ser utilizada.

Manual de uso

La aplicación ha sido creada para ser intuitiva y fácil de utilizar. Por esta razón, en el manual de uso no se utilizarán términos técnicos que pudieran resultar extraños o difíciles de entender para un usuario convencional.

INSTRUCCIONES GENERALES

Las instrucciones generales para la aplicación de escritorio son las siguientes:

1. Para acceder a la aplicación es necesario introducir el nombre de usuario y el código.
2. Para moverse por el contenido de una mesa, se debe arrastrar la barra lateral.
3. Para introducir un nuevo pedido en la tabla de stock se debe introducir correctamente la cantidad y la fecha de caducidad pulsando intro para validar la información. De lo contrario mostraría error y no grabaría el pedido.
4. Realizando una pulsación sobre un plato de cualquier mesa, se disparará un contador en el que se reflejará el tiempo de cocción de la preparación.
5. Realizando dos pulsaciones sobre un plato se dará por terminada la preparación, marcando el plato en color verde.
6. En el formulario de “añadir plato nuevo a la carta”, el botón buscar abrirá un buscador de archivos donde permitirá elegir fotos personalizadas de cada plato así añadirlas a la base de datos para posteriormente imprimirlas en la aplicación de móvil y así poder mostrarlas al cliente.

Las instrucciones generales para utilizar la aplicación Android son las siguientes:

1. Para acceder a la aplicación es necesario introducir el nombre de usuario y el código.
2. Para la selección de las mesas, clicar sobre el número de mesa deseado.
3. Personalizar los platos al gusto del cliente.
4. Pulsar el botón de validar para grabar el pedido en la base de datos.

CÓMO ENTRAR EN LA APLICACIÓN

Aplicación móvil

Al ejecutar la aplicación móvil el empleado tendrá que proceder a la autenticación.



Figura 6. Pantalla de autenticación de la aplicación android

El código que se ejecuta al pulsar el botón “Acceder”:

```

acceder.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        etiqueta.setVisibility(View.INVISIBLE);
        ejecutarServicio("http://192.168.42.139:80/ordenes/login.php", usuario.getText().toString(),
            contraseña.getText().toString());

    }
});

```

Dentro del código del botón “Acceder” se invoca el método “ejecutarServicio” que realiza una consulta en la base de datos para ver si el usuario y el código introducidos son correctos:

```

public void ejecutarServicio(String URL, final String usuario, final String contrasena){

    StringRequest stringRequest = new StringRequest(Request.Method.POST, URL, new
    Response.Listener<String>() {
        @Override
        public void onResponse(String response) {

            if(response.equals("1")){

                etiqueta.setVisibility(View.INVISIBLE);
                startMainActivity();
            } else {

                etiqueta.setVisibility(View.VISIBLE);
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            etiqueta.setVisibility(View.VISIBLE);
        }
    });
}

```

```

        Log.e("Volley Error", error.toString());
    }
}) {
    @Override
    protected Map<String, String> getParams(){
        Map<String,String> parametros= new HashMap<>();

        parametros.put("usuario", usuario);
        parametros.put("contrasena", contrasena);

        return parametros;
    }
};
RequestQueue requestQueue= Volley.newRequestQueue(this);

requestQueue.add(stringRequest);
}

```

Aplicación de escritorio

Al ejecutar la aplicación de escritorio, el usuario deberá identificarse en la pantalla de inicio:



Figura 7. Pantalla de inicio de la aplicación de escritorio

El código que se genera al pulsar el botón de acceso:

```

private void button_acceder_homeActionPerformed(java.awt.event.ActionEvent evt){
    Conector_bbdd conexion = new Conector_bbdd();

    if(conexion.comprobar_empleado(text_area_password.getText(), text_area_id_empleado.getText())){
        Home newHome = new Home();
        newHome.setVisible(true);
        this.setVisible(false);
    }else{
        label_incorrect_login.setVisible(true);
        label_incorrect_login.setText("ERROR");
    }
}
}

```

MANEJO DE PANTALLAS

Aplicación móvil

La correcta autenticación lleva a una nueva actividad donde el empleado elige el número de mesa que está atendiendo.

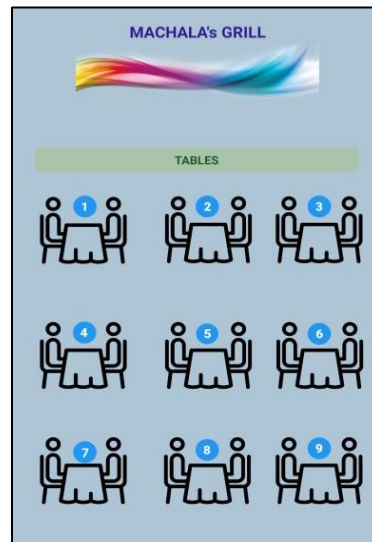


Figura 8. Pantalla de selección de mesa

Tras seleccionar la mesa, se accede a la pantalla de selección y personalización de platos. Aquí aparece un menú superior en el que se indica el número de mesa, las opciones de vaciar mesa o realizar pedido.

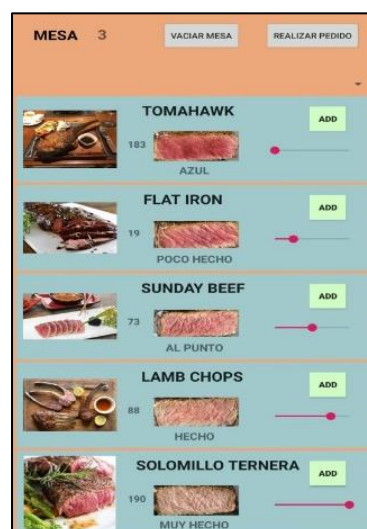


Figura 9. Pantalla de selección y personalización de platos.

A continuación está el spinner de platos a ordenar y el RecyclerView.

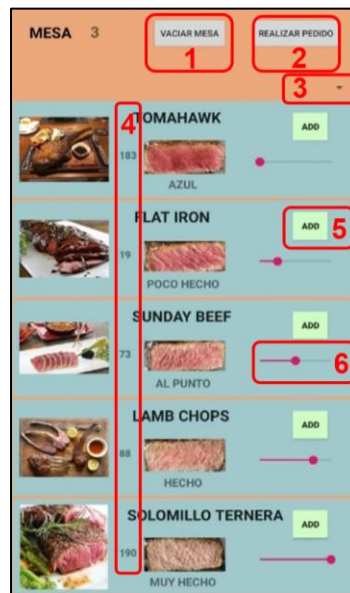


Figura 10. Funciones de los botones de la pantalla de selección y personalización de platos.

En la pantalla de selección y personalización de platos, aparecen 6 botones principales que ofrecen las distintas opciones de la pantalla. Se han numerado del 1 al 6 en la Figura 6.

A continuación se procede a explicar la funcionalidad de cada uno de los botones:

Botón 1: Pulsando este botón se vacía el spinner de platos listos para ordenar. Se vacía el ArrayList “platos_ordenados”.

```
boton_vaciar_mesa.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        platos_ordenados.clear();
        Toast.makeText(MesaActivity.this, "Pedido cancelado", Toast.LENGTH_SHORT).show();
    }
});
```

Botón 2: Mediante este botón se ejecuta el método “ejecutarServicioWeb” para confirmar los pedidos realizados. A través de la URL proporcionada se graba en la base de datos el pedido recogido y se muestra un mensaje Toast con el resultado obtenido. Se ejecuta el método para cada índice del Arraylist de “platos_ordenados”.

```
lanzar_mesa.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //recorro todos los platos
        for(int i = 0; i < platos_ordenados.size(); i++){
            ejecutarServicioWeb("http://192.168.42.139:80/ordenes/insertar_producto.php",
```

```

        platos_ordenados.get(i).getNombre(),
        platos_ordenados.get(i).getButter(),
        platos_ordenados.get(i).getPunto(),
        platos_ordenados.get(i).getSide(), atiende);
    }
    Toast.makeText(MesaActivity.this, "CORRECTO", Toast.LENGTH_SHORT).show();
    platos_ordenados.clear();
}
});

```

El método recibe la URL y las propiedades de cada índice y lanza una petición al back end, que devuelve el estado de la consulta tras ejecutarla en el servidor SQL.

```

private void ejecutarServicioWeb(String URL, final String nombre, final String butter, final String
    punto, final String side, final String atiende){
    StringRequest stringRequest = new StringRequest(Request.Method.POST, URL, new
        Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            //capturo stock existente para cada ingrediente
            capturar_stock("http://192.168.42.139:80/ordenes/captar_stock.php");
        }
    }, new Response.ErrorListener() { @Override..... }
    }) {
        @Override //en parametros se agregan los datos
        protected Map<String, String> getParams(){.....}
    };
    RequestQueue requestQueue= Volley.newRequestQueue(this);

    requestQueue.add(stringRequest);
}

```

Botón 3: Spinner con los platos listos para ordenar. Este spinner lista todos los platos personalizados y ofrece una visión global de la mesa y la posibilidad de pulsar sobre un elemento del listado para borrarlo individualmente.

```

class AdaptadorMesa extends ArrayAdapter<OrdenPlato> {

    public AdaptadorMesa(@NonNull Context context, @NonNull List<OrdenPlato> objects) {
        super(context, R.layout.mesa_layout, objects);
    }
    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup
    parent) {
        return getDropDownView(position, convertView, parent);
    }
}

```



```

@Override
public View getDropDownView(int position, @Nullable View convertView, @NonNull
ViewGroup parent) {

    OrdenPlato ordenado = getItem(position);

    View view;
    if (convertView == null) {
        view = LayoutInflater.from(getContext()).inflate(R.layout.mesa_layout, null);
    } else {
        view = convertView;
    }
    TextView textNombre = view.findViewById(R.id.text_nombre_plato);
    TextView textButter = view.findViewById(R.id.texto_butter);
    TextView textPunto = view.findViewById(R.id.texto_punto);
    TextView idSide = view.findViewById(R.id.texto_side);

    textNombre.setText(ordenado.getNombre());
    textButter.setText(ordenado.getButter());
    textPunto.setText(ordenado.getPunto());
    idSide.setText(ordenado.getSide());

    return view;
}
}

```

Botón 4: Esta etiqueta actualiza el stock al iniciar la actividad o tras cada grabado de pedido de cada producto, para evitar la realización de pedidos con existencias insuficientes.

```

public void capturar_stock(String URL){
    JSONObjectRequest jsonRequest = new JSONObjectRequest(Request.Method.POST, URL, null,
    new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response){

            ArrayList<Integer>cantidades_stock = new ArrayList<>();
            ArrayList<Integer>indeicesDB = new ArrayList<>();
            JSONArray jsonStock;

            //trato de guardar en jsonStock los valores devueltos por "Response" buscandolo por "stock"
            try { ..... } catch (JSONException e) {
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        ArrayList<Integer>indeicesDB = new ArrayList<>();

```

```

ArrayList<Integer>cantidades_stock = new ArrayList<>();

//en caso de error sigo con la ejecución del programa donde Done será falso
call_back_stock(cantidades_stock, indicesDB, false);
Log.e("Server Error", error.toString());
});
RequestQueue requestQueue = Volley.newRequestQueue(this);
requestQueue.add(jsonRequest);
}

```

Botón 5: Botón de confirmación del punto de las carnes. Abre ventana Dialog para completar plato con salsa y/o guarnición. Mediante el botón “ok” se guarda el plato personalizado en el spinner de platos personalizados.

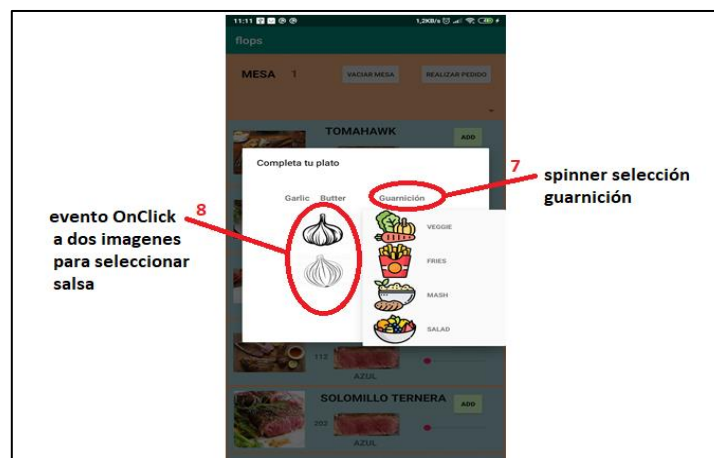


Figura 11. Ventana Dialog para elegir salsa y/o guarnición.

Spinner de elección de la guarnición:

```

public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    switch(position){
        case 0:    temporal = "Vegetables";
                    break;
        case 1:    temporal = "Fries";
                    break;
        case 2:    temporal = "Mash Potatoes";
                    break;
        case 3:    temporal = "Salad";
                    break;
    }
}

```

Determinación de la salsa elegida:

```

garlic_pic.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

```

```

        eleccion_salsa = "garlic butter";

        texto_salsa.setText("Garlic");

    }

});
onion_pic.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        eleccion_salsa = "onion butter";

        texto_salsa.setText("Onion");

    }

});

```

Botón 6: Mediante este seekbar, se actualiza foto y el texto referente al punto de cocción de la carne.

```

private void updateProgress(int progress) {

    if(progress == 0) {

        holder.punto_carne.setText("AZUL");

        holder.color_coccion.setImageResource(R.drawable.carne1);

    }else if(progress == 1){

        holder.punto_carne.setText("POCO HECHO");

        holder.color_coccion.setImageResource(R.drawable.carne2);

    }else if(progress == 2){

        holder.punto_carne.setText("AL PUNTO");

        holder.color_coccion.setImageResource(R.drawable.carne3);

    }else if(progress == 3){

        holder.punto_carne.setText("HECHO");

        holder.color_coccion.setImageResource(R.drawable.carne4);

    }else {

        holder.punto_carne.setText("MUY HECHO");

        holder.color_coccion.setImageResource(R.drawable.carne5);

    }

}
}

```

Aplicación de escritorio

Tras realizar la identificación la siguiente pantalla ofrece tres posibilidades:



Figura 12. Pantalla de inicio de la aplicación de escritorio

GO LIVE: Mediante este botón se abre la pantalla de ordenes en vivo dónde aparecen las ordenes pendientes o, en su defecto, se queda a la espera de nuevos pedidos.

//botón que abre el JFrame de LiveForm

```
private void botonGoLiveActionPerformed(java.awt.event.ActionEvent evt) {  
    boton_historial.setVisible(true);  
    newLive = new LiveForm(this);  
    newLive.setVisible(true);  
    this.setVisible(false); }  

```

HISTORIAL: Este botón solo aparece si se accede a la opción Go Live y mostrará mediante una tabla las órdenes transcurridas. En caso de no haber ninguna orden, el botón queda deshabilitado.

Numero Mesa	Nombre Plato	Punto Cocción	Guarnición	Salsa Butter	Veces Repetido	Atiende en Mesa	Hora Pedido
6	LAMB CHOPS	AZUL	Vegetables	/	1	admin	12:37
5	SUNDAY BEEF	AZUL	Vegetables	garlic butt	1	admin	12:37
5	SUNDAY BEEF	AZUL	Vegetables	/	1	admin	12:37
5	SOLOMILLO TE...	AZUL	Vegetables	onion butte	1	admin	12:37
7	SOLOMILLO TE...	AZUL	Salad	garlic butt	1	admin	12:37
7	FLAT IRON	AL PUNTO	Fries	garlic butt	1	admin	12:37
7	LAMB CHOPS	HECHO	Mash Potatoes	onion butt	1	admin	12:38
7	LAMB CHOPS	AZUL	Fries	onion butt	1	admin	12:38
7	SOLOMILLO TE...	AZUL	Vegetables	onion butt	1	admin	12:38
3	SOLOMILLO TE...	MUY HECHO	Vegetables	garlic but	1	admin	12:38
3	FLAT IRON	AL PUNTO	Salad	onion butt	1	admin	12:38
1	TOMAHAWK	AZUL	Fries	garlic but	1	admin	12:38
1	COSTILLAR GRI...	AZUL	Mash Potatoes	onion butt	1	admin	12:38

Figura 13. Pantalla de historial de órdenes

El código que genera la tabla de historial:

```
private void boton_historialActionPerformed(java.awt.event.ActionEvent evt)

    ArrayList<Productos> ordenados = new ArrayList<>();

    if(newLive.getProductos_CSV().size() > 0){

        ordenados = newLive.getProductos_CSV()

        this.setVisible(false);

        HistorialDia history = new HistorialDia(this, ordenados);

        history.setVisible(true); }}
```

A continuación se muestra un ejemplo de tabla de historial:

Tabla 1. Ejemplo de historial de comandas

Numero Mesa	Nombre	Punto	Guarnición	Salsa	Cantidad	Atiende	Hora pedido
5	SALCHICHAS	AZUL	Vegetables	/	1	admin	13:50
5	CHULETAS CORDERO	AZUL	Vegetables	/	1	admin	13:54
6	SUNDAY BEEF	AZUL	Vegetables	/	2	admin	13:50
6	COSTILLAR GRILL	AZUL	Vegetables	/	1	admin	13:50
7	SUNDAY BEEF	AZUL	Vegetables	/	2	admin	13:50
1	SUNDAY BEEF	AZUL	Vegetables	/	1	admin	15:50
2	FLAT IRON	AZUL	Vegetables	/	1	admin	15:50
2	LAMB CHOPS	AZUL	Vegetables	/	1	admin	15:53

ADMINISTRACIÓN: Al acceder a esta sección se mostrará el stock actual y tendrá en su parte inferior un botón para insertar otros pedidos. También dispondrá de un botón que abrirá otra pantalla donde se podrán dar de baja o añadir nuevos platos a la carta.

STOCK ACTUAL		
Nombre	Cantidad	Fecha_Cad
TOMAHAWK	179	20/02/2020
FLAT IRON	17	12/09/2018
SUNDAY BEEF	69	25/09/2020
LAMB CHOPS	79	12/02/2020
SOLOMILLO TERNERA	189	28/02/2021
ENGLISH BFAST	40	20/02/2020
COSTILLAR BBQ	40	20/02/2020
CHURRASCO	76	20/12/2021
PS BURGER	45	20/04/2020
SALCHICHAS	182	20/02/2020
BEEF BURGER	387	20/02/2020
CHULETAS CORDERO	0	20/02/2020

Figura 14. Pantalla de visualización del stock actual

El código que abre la pantalla de visualización del stock:

```
private void boton_stockActionPerformed(java.awt.event.ActionEvent evt)

//crea nuevo stockForm y le pasa este para poder volver a la misma sesión

StockForm newStock = new StockForm(this);

newStock.setVisible(true)

this.setVisible(false); }
```

Solución de posibles problemas

El usuario tendrá una pequeña guía orientativa sobre los errores con los que se pudiera encontrar durante el uso diario de la aplicación.

Tabla 2. Posibles problemas y soluciones

PROBLEMAS	SOLUCIONES
Error de conexión a la base de datos	XAMPP sin inicializar o cambio en la dirección IP del ordenador. Consultar problema con el desarrollador de la aplicación
Descuadre del tamaño de las ventanas de la aplicación	No utilizar un monitor con una resolución distinta a FULL HD. En caso de no tener esta opción, consultar el problema con el desarrollador de la aplicación.
Botón historial visible pero deshabilitado	Significa que en el día presente no se ha lanzado ninguna orden. El botón se habilitará con la primera comanda.
Fallo inesperado del sistema (corte de luz, cuelgue de la aplicación, cierre por error...)	Las órdenes pendientes se vuelven a imprimir en el mismo orden que estaban.
Stock insuficiente	Queda deshabilitado el botón de añadir plato al pedido.