

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

***CHEF MANAGER***

Icono

Descripción generada automáticamente

Trabajo de Fin de Grado

Tutor: Antonio Carmona Lara

Alumno: David Chavarria Paccha

**ÍNDICE**

1. Resumen ……………………………………………………………………
   1. Justificación ………………………………………………………
   2. Análisis de mercado ……………………………………………
2. Metodología ……………………………………………………………….
3. RTFP …………………………………………………………………
4. Diagramas …………………………………………………………………
   1. . Diagrama Base de datos ………………………………………….
   2. Diagrama Clases …………………………………………………..
   3. Diagrama interfaces ………………………………………………..
5. Maqueta interfaces …………………………………………………….
6. Casos de uso …………………………………………………………….
7. Tecnologías utilizadas ……………………………………………….
8. GIT ………………………………………………………………………….
9. Diagrama Gantt ……………………………………………………….
10. Bibliografía …………………………………………………………

**Resumen**

Chef Manager es una aplicación para la correcta gestión de restaurantes cuyo propósito es simplificar y optimizar las operaciones diarias en la industria de la hostelería. La aplicación sustenta diferentes aspectos de la gestión de restaurantes, como la administración de proveedores, empleados, inventario y de los beneficios/pérdidas. El objetivo de Chef Manager es ofrecer a sus usuarios una herramienta eficiente para reducir las tareas administrativas, mejorar la comunicación y optimizar la gestión de su negocio. Entre sus principales características se encuentran:

1. *Administración de proveedores*: Permite a los usuarios gestionar sus relaciones con los proveedores, controlar el historial de compras, realizar el seguimientos de los productos y albergar la información de todos los proveedores de una manera organizada,
2. *Gestión de empleados:* Apartado en el cuál se administra el personal. Se almacena toda la información de cada empleado como el cargo, la nómina, turnos/horarios y su rendimiento.
3. *Control del inventario:* Ayuda para que los usuarios realicen un seguimiento de las existencias, establecer niveles de reordenamiento y minimizar el desperdicio de alimentos.
4. *Caja:* En esta sección se administrará todo el dinero obtenido en la caja, tanto diario, como semanalmente.
5. *Calendario:* Página que muestra el horario mensual que tendrá el usuario. Días festivos, no lectivos, periocidad de los proveedores etc.
6. *Notas:* Breve apartado para escribir cualquier anotación sobre la empresa que se quiera hacer.

En resumen, Chef Manager está dirigido a propietarios y gerentes de restaurantes que buscan mejorar su eficiencia y rentabilidad ya que es una herramienta esencial que combina una interfaz intuitiva y funciones avanzadas.

**Abstract**

Chef Manager is an application for the correct management of restaurants whose purpose is to simplify and optimize daily operations in the hospitality industry. The application supports different aspects of restaurant management, such as supplier, employee, inventory, and profit/loss management. The objective of Chef Manager is to offer its users an efficient tool to reduce administrative tasks, improve communication and optimize the management of their business. Among its main features are:

1. *Supplier Management:* Allows users to manage their relationships with suppliers, control purchasing history, track products and house information on all suppliers in an organized way,

2. *Employee management:* Section in which personnel is managed. All information about each employee is stored, such as position, payroll, shifts/schedules, and performance.

3. *Inventory Control:* Helps users track stock, set reorder levels, and minimize food waste.

4. *Cash:* In this section all the money obtained in the cash register will be managed, both daily and weekly.

5. *Calendar:* Page that shows the monthly schedule that the user will have. Holidays, non-school days, frequency of suppliers, etc.

6. *Notes*: Brief section to write any notes about the company that you want to make.

In short, Chef Manager is aimed at restaurant owners and managers looking to improve their efficiency and profitability as it is an essential tool that combines an intuitive interface and advanced features.

**Justificación**

La gestión eficaz de un restaurante es fundamental para su éxito y rentabilidad en la industria hotelera, la cual es crucial en la economía global. La app Chef Manager satisface las necesidades del sector gastronómico al proporcionar una solución completa para administrar restaurantes, proveedores, personal, inventario y reservas. Este estudio se fundamenta en la importancia de analizar si la aplicación Chef Manager es eficaz para mejorar la gestión de restaurantes. Se busca averiguar si la app puede disminuir el tiempo empleado en labores administrativas, mejorar la comunicación con proveedores y empleados, y optimizar la gestión de inventarios y reservas mediante una investigación detallada.  
La investigación se enfoca en examinar las características y funciones de Chef Manager, además de recolectar comentarios y vivencias de usuarios reales. El enfoque consiste en realizar entrevistas, encuestas y análisis de datos para medir cómo la aplicación afecta la eficacia y rentabilidad de los restaurantes.  
Los hallazgos de este análisis pueden ofrecer datos importantes acerca de la efectividad de la plataforma Chef Manager y su capacidad de revolucionar la administración en restaurantes. Además, los descubrimientos podrían ser útiles para propietarios y gerentes de restaurantes, cadenas de restaurantes y empresas de catering al momento de seleccionar herramientas de gestión eficaces y adecuadas para sus requerimientos.  
En resumen, la razón por la que se realiza este estudio es la importancia de la gestión efectiva en la industria de la hostelería y la capacidad de Chef Manager para aumentar la eficiencia y rentabilidad de los restaurantes.

**Análisis de mercado**

Actualmente, el mercado ofrece diversas soluciones de software para la gestión de restaurantes. Sin embargo, muchas de estas aplicaciones se centran en aspectos más generales y no ofrecen una solución concreta. Además, suelen ser complejos y no siempre fáciles de usar, lo que dificulta su adopción por parte del personal con distintos niveles de habilidades tecnológicas. Las aplicaciones existentes suelen ser eficaces para gestionar reservas, pedidos y pagos, pero muchas no integran eficazmente el control de inventario, la gestión del personal y de los proveedores, ni ofrecen una visión clara de la salud financiera del restaurante. El vacío que busca llenar nuestra aplicación es la integración de todas estas funciones en una interfaz intuitiva y accesible.

**Metodología**

La metodología que utilicé para este proyecto es el de “Metodología Cascada” el cual toma un enfoque estructurado y secuencial para el desarrollo de aplicaciones que se basa en el concepto de fases discretas y lineales. Cada fase se completa antes de pasar a la siguiente, y el proceso no se vuelve a visitar a menos que se detecten errores o se requieran cambios. En el proyecto Chef Manager, las etapas de la metodología Cascada incluyen:

Diagrama

Descripción generada automáticamente

**RFTP**

Requisitos Funcionales: El software debe permitir al usuario registrar un restaurante e inmediatamente registrar un usuario para crear una cuenta usuario-restaurante. También debe permitir al usuario iniciar sesión. Después debe poder crear, editar, eliminar y ver Empleados, Proveedores, Productos y Notas, es decir, personalizar su aplicación a su gusto.

Requisitos no funcionales: El software debe validar todos los campos para no crear conflictos con la base de datos ni con la interfaz. Tiene que ser intuitivo y fácil de entender. Debe ser compatible con diferentes dispositivos.

Funcionalidades: - Módulo de inicio de sesión: Permite al usuario iniciar sesión en el sistema. - Módulo de registro: Permite al usuario crear una cuenta y un restaurante en el sistema. -Módulo de CRUD: permite personalizar al usuario los datos de su restaurante (crear, leer, editar y eliminar). -Módulo de búsqueda: en cada vista que tenga una lista se incluirá un buscador para facilitar la experiencia.

Tareas: -Diseñar la interfaz de bienvenida, el módulo de registro e inicio de sesión. -Integrar el módulo de validaciones para ambos registros -Diseñar la interfaz de administrador e implementar el módulo de CRUD para todos las clases (Empleados, Productos, Usuarios,Restaurantes…) de la VistaAdmin. Diseñar la VistaPrincipal. -Implementar la conexión con la API y mostrarla en la vista. -Desarrollar todas las interfaces de las clases(Empleados, Productos, Notas …). -Implementar las funcionalidades personalizadas de cada clase. -Implementar las validaciones en todas las clases. -Integrar el módulo de búsqueda en cada una.

Pruebas:

( VistaLogin, VistaUserRegister, VistaRestauranteRegister, VistaAdmin )

Validaciones

* Validar que los campos de registro tanto de usuario como restaurante (nombre, email, contraseña, logo etc.) acepten los tipos de datos correctos y den error si se ingresan datos inválidos.
* Probar que el registro se complete exitosamente con datos válidos y que el usuario quede autenticado
* Verificar que el usuario registrado pueda iniciar sesión con sus credenciales correctas
* Comprobar que el login falle si se ingresan credenciales incorrectas (usuario o contraseña erróneos)
* Probar que el usuario pueda cerrar sesión exitosamente
* Validar que no se permita el acceso a usuarios no autenticados a la vista administrador.

Pruebas de integración

* Verificar que el registro e inicio de sesión funcionen correctamente con la base de datos.
* Probar que el usuario autenticado con sus credenciales pueda acceder a las funcionalidades permitidas según su id.

( Navegación entre clases y en la misma clase )

* Validar que al navegar entre vistas, la transición sea suave y sin errores visuales o de funcionamiento
* Comprobar que al cambiar de una vista a otra, la información se cargue correctamente y se muestre de manera adecuada
* Verificar que los elementos de navegación, como botones o enlaces, funcionen correctamente al dirigir al usuario a las vistas correspondientes
* Probar la consistencia en el diseño y estilo de las vistas para mantener una apariencia uniforme en toda la aplicación
* Validar que al regresar a vistas anteriores, la aplicación mantenga el estado adecuado y no se pierda información importante

( Vista Principal y clases subyacentes donde haya CRUD)

Pruebas de Creación (Create)

* Verificar que se puedan crear nuevos registros en la base de datos correctamente.
* Validar que todos los campos obligatorios se completen al momento de crear un nuevo registro.
* Comprobar que se genere un identificador único para cada nuevo registro creado.

Pruebas de Lectura (Read)

* Verificar que se puedan recuperar los registros existentes de la base de datos de forma correcta.
* Comprobar que se puedan filtrar los registros según criterios específicos.
* Validar que se puedan ordenar los registros de acuerdo con diferentes campos.

Pruebas de Actualización (Update)

* Verificar que se puedan modificar los registros existentes de manera precisa.
* Comprobar que los cambios realizados en un registro se reflejen correctamente en la base de datos.
* Validar que se puedan actualizar múltiples registros a la vez si es necesario.

Pruebas de Eliminación (Delete)

* Verificar que se puedan eliminar registros de la base de datos de forma segura.
* Comprobar que al eliminar un registro, se actualicen correctamente las relaciones con otros registros.
* Validar que se implementen medidas de seguridad para prevenir eliminaciones accidentales.

Pruebas de Integración

* Verificar que las operaciones CRUD se integren correctamente con otros módulos de la aplicación.
* Validar que se mantenga la consistencia de los datos al realizar operaciones CRUD en diferentes partes de la aplicación.

**Diagrama de clases**

Diagrama

Descripción generada automáticamente

**Diagrama de base de datos**

Diagrama

Descripción generada automáticamente

**Casos de uso**

Una vez que tenemos una idea clara de qué requisitos y necesidades debería cubrir la aplicación, es hora de diseñar sus casos de uso.

Un caso de uso es una descripción de los pasos o actividades que se deben realizar para ejecutar un proceso. Los personajes o entidades que participan en un caso de uso se denominan actores. En el contexto del desarrollo de software, un caso de uso es un conjunto de interacciones que se desarrollan entre una aplicación y los usuarios en respuesta a eventos iniciados por los usuarios en la propia aplicación. Los diagramas de casos de uso se utilizan para especificar la comunicación y el comportamiento de un sistema a través de sus interacciones con los usuarios y otros sistemas. La aplicación consta de varios casos de uso diferentes. Entre todos estos, la mayoría son simples debido al requisito de que debería ser intuitiva la aplicación . Un ejemplo sería: botón "Cerrar sesión". Esto simplemente significa que el usuario cierra la vista principal y le devuelve a la vista del login. O también los diferentes registros como los empleados o los proveedores.

**Bibliografía**

Upload images info: <https://www.youtube.com/watch?v=jlQr7aTXMds>

Microsoft Documentation: <https://learn.microsoft.com/es> es/dotnet/maui/?view=net-maui-8.0