Aprendiz: Rafael Dario Escalante Sandoval

Equipo de proyecto:

Saray Acosta

Cristian Mosquera Rodríguez

Danny Alexander Minota Soto

Instructor: Andrés Rubiano Cucarian

ADSO (2977466)

informe de entregables para el proyecto de desarrollo de softwareGA4220501095-AA2-EV02.

Contents

Informe de Análisis del Proyecto "Mitady"

1. Introducción

El proyecto "Mitady" es una cafetería que ofrece una variedad de comidas y bebidas con servicio a domicilio. El objetivo del software a desarrollar es facilitar la gestión de pedidos, clientes, productos, pagos y entregas, optimizando la experiencia del usuario y mejorando la eficiencia operativa.

2. Objetivos del Proyecto

- Proporcionar una plataforma fácil de usar para que los clientes realicen pedidos en línea.
- Gestionar la base de datos de clientes, productos y pedidos de manera eficiente.
- Facilitar el seguimiento de pagos y entregas.
- Mejorar la comunicación entre el personal de la cafetería y los repartidores.

3. Características del Software a Diseñar

3.1. Interfaz de Usuario

- **Diseño Atractivo**: Interfaz intuitiva y visualmente agradable para los usuarios.
- **Compatibilidad Móvil**: Adaptabilidad para su uso en dispositivos móviles y tablets.
- **Registro y Autenticación de Usuarios**: Posibilidad de que los clientes creen cuentas para gestionar sus pedidos y pagos.

3.2. Gestión de Clientes

- **Registro de Clientes**: Registro de información básica (nombre, apellido, teléfono, dirección, correo electrónico).
- **Historial de Pedidos**: Visualización del historial de pedidos anteriores de cada cliente.
- **Gestión de Preferencias**: Permitir a los clientes guardar sus productos favoritos y métodos de pago preferidos.

3.3. Gestión de Productos

- **Catálogo de Productos**: Listado de todos los productos disponibles (comidas y bebidas) con descripciones y precios.

- **Control de Inventario**: Seguimiento de la disponibilidad de productos y alertas para reabastecimiento.
- **Actualización de Productos**: Posibilidad de añadir, modificar o eliminar productos fácilmente.

3.4. Gestión de Pedidos

- **Creación de Pedidos**: Proceso sencillo para que los clientes agreguen productos al carrito y realicen pedidos.
- **Seguimiento de Pedidos**: Posibilidad de que los clientes sigan el estado de su pedido (pendiente, en preparación, en entrega, entregado).
- **Notificaciones**: Alertas automáticas a los clientes sobre el estado de su pedido.

3.5. Gestión de Pagos

- **Métodos de Pago**: Integración de diferentes métodos de pago, incluyendo Nequi, tarjetas de crédito/débito y efectivo.
- **Historial de Pagos**: Registro de todos los pagos realizados por los clientes con detalles sobre el método y el monto.

3.6. Gestión de Repartidores

- **Asignación de Entregas**: Asignación automática de pedidos a repartidores disponibles.
- **Seguimiento de Entregas**: Registro de entregas realizadas y estado de cada entrega.

3.7. Informes y Análisis

- **Estadísticas de Ventas**: Informes sobre las ventas diarias, semanales y mensuales.
- **Análisis de Productos**: Reportes sobre los productos más vendidos y los menos vendidos.
- **Feedback de Clientes**: Recopilación de opiniones de los clientes para mejorar la calidad del servicio.

Tecnologías Recomendadas

- **Lenguajes de Programación**: JavaScript (para el frontend), PHP o Python (para el backend).

- **Base de Datos**: MySQL para gestionar la información.
- **Frameworks**: React o Angular para el frontend; Flask o Laravel para el backend.
- **Métodos de Pago**: Integración con API de Nequi y otros métodos de pago.

. Frontend (Interfaz de Usuario)

- **Lenguaje de Programación**: JavaScript, HTML, y CSS.
- **Frameworks y Librerías**:
    - **React** o **Vue.js**: Frameworks de JavaScript para construir una interfaz de usuario dinámica y responsiva. Estos frameworks permiten crear componentes reutilizables y gestionar el estado de la aplicación eficientemente.
    - **Bootstrap** o **Tailwind CSS**: Frameworks de CSS para diseñar interfaces atractivas y adaptables en dispositivos móviles y de escritorio, esenciales para una experiencia fluida en la aplicación.
- **Herramientas de Testing y Desarrollo**:
    - **Jest** y **React Testing Library** (para React) o **Vue Test Utils** (para Vue): Herramientas para pruebas unitarias y de componentes, asegurando la calidad del código en el frontend.
    - **Webpack** o **Vite**: Para optimizar el empaquetado de los recursos del frontend, mejorando el rendimiento y tiempos de carga.

2. Backend (Lógica del Negocio y API)

- **Lenguaje de Programación**: Node.js o Python.
- **Frameworks**:
    - **Express.js** (para Node.js) o **Django**/ **Flask** (para Python): Frameworks que facilitan la creación de APIs RESTful y gestionan las operaciones CRUD (crear, leer, actualizar y eliminar) para los datos de clientes, productos, pedidos y entregas.
- **Autenticación y Autorización**:
    - **JWT (JSON Web Tokens)** o **OAuth2** para autenticar usuarios de forma segura y manejar permisos, esencial en el acceso a funcionalidades de usuario y administración.

3. Base de Datos

- **Base de Datos Relacional (SQL)**:
    - **MySQL** o **PostgreSQL**: Bases de datos relacionales que manejan la estructura de datos con integridad y escalabilidad. Su estructura basada en tablas es ideal para relaciones entre clientes, pedidos y productos.
    - **ORM (Object-Relational Mapping)**: **Sequelize** para Node.js o **SQLAlchemy** para Python, que facilita la interacción entre el código orientado a objetos y la base de datos, haciendo el desarrollo más ágil y organizado.

4. Plataforma de Pagos

- **Integración con API de Pago**:
    - **Nequi API** y **Pasarelas de Pago**: Integrar la API de Nequi para pagos en Colombia y, opcionalmente, un proveedor de pagos como **Stripe** o **PayU** para manejar otros métodos de pago, mejorando la accesibilidad del sistema.

5. Plataformas para Notificaciones y Mensajería

- **Twilio** o **Firebase Cloud Messaging (FCM)**: Para enviar notificaciones a los clientes sobre el estado de sus pedidos, además de alertas al personal para coordinar entregas y cambios en el estado del inventario.

6. Servicios en la Nube e Infraestructura

- **Plataforma de Alojamiento**:
    - **Heroku**, **AWS (Amazon Web Services)** o **DigitalOcean**: Proporcionan servidores escalables para alojar el backend y gestionar las bases de datos. AWS también ofrece servicios como RDS para bases de datos administradas y S3 para almacenamiento de archivos (como imágenes de productos).
- **CDN (Red de Entrega de Contenidos)**:
    - **Cloudflare** o **AWS CloudFront**: Aseguran una entrega rápida de contenido estático (imágenes, archivos de estilo) en diferentes ubicaciones geográficas, mejorando el rendimiento de la aplicación.

7. Pruebas y Automatización

- **CI/CD (Integración Continua y Despliegue Continuo)**:

- **GitHub Actions**, **Travis CI** o **Jenkins**: Configurar un pipeline de CI/CD para probar y desplegar automáticamente los cambios en el software, garantizando una entrega continua y una mejor gestión de versiones.

8. Monitoreo y Seguridad

- **Monitoreo y Logs**:
  - **Sentry** o **New Relic**: Para supervisar errores, rendimiento y errores en tiempo real, y mejorar el mantenimiento de la aplicación.
- **Seguridad**:
  - **Cifrado SSL/TLS** para asegurar la comunicación de datos.
  - **Escaneo de Vulnerabilidades**: Integrar herramientas como **Dependabot** (para Node.js) o **Bandit** (para Python) en el proceso de CI/CD para detectar vulnerabilidades en dependencias.

Entregables de Diseño para "Mitady"

1. **Modelo de Dominio**
   - **Descripción**: Representación visual de las entidades del dominio y sus relaciones. Incluye diagramas que muestran las clases principales, sus atributos, y las relaciones entre ellas.
   - **Ejemplo de Clases**:
     - Cliente
     - Producto
     - Pedido
     - DetallePedido
     - Pago
     - Repartidor
     - Entrega
2. **Diagramas de Clases**
   - **Descripción**: Diagramas que representan las clases, sus atributos, métodos y las relaciones (asociación, herencia, composición) entre ellas.

- **Detalles a Incluir**:
  - Atributos y tipos de datos.
  - Métodos públicos y privados.
  - Relaciones (1 a N, N a N) entre clases.

3. **Diagramas de Secuencia**
   - **Descripción**: Diagramas que muestran la interacción entre objetos a lo largo del tiempo para un caso de uso específico. Ayudan a entender cómo se comunican las clases durante la ejecución de un proceso.
   - **Ejemplo**: Proceso de realizar un pedido, donde se muestran los pasos desde que el cliente selecciona un producto hasta que se confirma el pago.
4. **Diagrama de Casos de Uso**
   - **Descripción**: Diagrama que representa las interacciones entre actores (usuarios, sistemas) y el sistema. Define las funcionalidades del sistema desde la perspectiva del usuario.
   - **Ejemplos de Casos de Uso**:
     - Realizar un pedido.
     - Pagar un pedido.
     - Asignar un repartidor.
     - Consultar historial de pedidos.
5. **Definición de Interfaces**
   - **Descripción**: Especificaciones de las interfaces de usuario y de programación (API). Incluyen detalles sobre la navegación, elementos visuales y cómo interactuarán los usuarios con el sistema.
   - **Ejemplo**: Mockups o wireframes de la interfaz de usuario, que muestren cómo se verá la página de inicio, la página de catálogo de productos, el carrito de compras, etc.
6. **Patrones de Diseño**
   - **Descripción**: Documentación sobre los patrones de diseño utilizados en el sistema, como el patrón MVC (Modelo-Vista-Controlador), Singleton, Factory, etc. Indicar cómo se aplican en el contexto del proyecto.

- **Ejemplos de Aplicación**:
    - Uso de Singleton para la gestión de la conexión a la base de datos.
    - Implementación de Factory para la creación de diferentes tipos de productos.

7. **Documentación Técnica**
    - **Descripción**: Manual de usuario y documentación técnica para desarrolladores, que incluya detalles sobre la implementación, configuración y uso del sistema.
    - **Secciones a Incluir**:
        - Guía de instalación.
        - Descripción de la arquitectura del sistema.
        - Ejemplos de cómo utilizar las clases y métodos.
8. **Plan de Pruebas**
    - **Descripción**: Estrategia de pruebas que defina cómo se evaluará el software, incluyendo pruebas unitarias, pruebas de integración y pruebas de aceptación.
    - **Componentes a Considerar**:
        - Casos de prueba para cada clase y método.
        - Estrategias para realizar pruebas de usuario y feedback.

5. Conclusión

El software por desarrollar para la cafetería "Mitady" tiene como objetivo mejorar la experiencia del cliente y la eficiencia operativa. Con un enfoque en la usabilidad y funcionalidades clave, el proyecto busca ser una solución integral para la gestión de pedidos de comida y bebida a domicilio. La implementación adecuada de estas características permitirá que "Mitady" se destaque en el mercado y brinde un servicio de calidad a sus clientes.