UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Facultad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Análisis y Diseño de un Sistema de Programación

Materia : Programación Orientada a Objetos

Nombre	**Matricula**	**Carrera**
Alfonso Alejandro Virgen Delgado	2177999	ITS
Angel Martin Gonzalez Esquivel	2096529	ITS
Christian Sebastian Moreno Gonzalez	1950199	ITS
Christian Alberto Rangel Hernández	2027868	IAS

Hora : N3

Grupo : 017

Docente : Ana Karen Antopia Barrón

Fecha : 23 de Febrero del 2025

1. Revisión Teórica del Contenido Conceptual

Definición y Características de los sistemas de Programación:

Es un conjunto de herramientas y recursos que proporciona un entorno para escribir, depurar, compilar y ejecutar programas informáticos. Incluye tanto el lenguaje de programación como el entorno de desarrollo integrado (IDE), compiladores, depuradores y otros recursos necesarios para la creación de un software.

Características:

1. **Lenguaje de programación:** el lenguaje utilizado para escribir el código fuente del software, como Java, C + +, Python, etc.
2. **Entorno de desarrollo integrado (IDE):** un software que proporciona una interfaz de usuario amigable para escribir, depurar y compilar código. Ejemplos incluyen Visual Studio, Eclipse y PyCharm.
3. **Compilador/Intérprete:** herramientas que traducen el código fuente en código de máquina que puede ser ejecutado por la computadora.
4. **Depurador:** Un programa que permite a los desarrolladores encontrar y corregir errores en el código fuente.
5. **Gestión de versiones:** Sistemas que permiten controlar las diferentes versiones del código fuente y colaborar con otros desarrolladores, GthHub es un ejemplo popular.
6. **Biblioteca y Frameworks:** Conjunto de código preescrito para que los desarrolladores puedan utilizar las herramientas del sistema de programación.
7. **Documentación:** Recursos y manuales que ayudan a los desarrolladores a entender y utilizar las herramientas del sistema de programación.
8. **Soporte de Plataformas:** Algunos sistemas de programación son multiplataforma, lo que significa que pueden ser utilizados en diferentes sistemas operativos como Windows, macOS y Linux.

Modelos y metodologías de desarrollo aplicadas en análisis y diseño de software:

MODELO	DESCRIPCIÓN	FASES	VENTAJAS	DESVENTAJAS
Modelo en Cascada	Es un enfoque lineal y secuencial donde cada fase debe completarse antes de que la siguiente comience.	Requisitos, Diseño, Implementación, Pruebas, Despliegue, Mantenimiento.	Simple y fácil de entender, adecuado para proyectos con requisitos bien definidos.	Poco flexible, difícil de realizar cambios una vez que una fase se ha completado.
Modelo Incremental	El proyecto se desarrolla en pequeñas iteraciones o incrementos, cada uno de los cuales añade funcionalidad al sistema final.	Similar al modelo en cascada pero repetido para cada incremento.	Permite entregas parciales y tempranas del sistema, fácil de realizar cambios.	Puede requerir más planificación y gestión.
Modelo en Espiral	Combina elementos del modelo en cascada y del modelo iterativo, enfatizando la gestión de riesgos.	Planificación, Análisis de Riesgos, Ingeniería, Evaluación del Cliente.	Enfocado en la gestión de riesgos, adecuado para proyectos grandes y complejos.	Más complejo y costoso, requiere experiencia en gestión de riesgos.
Modelo V	Extensión del modelo en cascada con una fuerte énfasis en la verificación y validación.	Similar al modelo en cascada pero con actividades de prueba para cada fase de desarrollo.	Mejora la calidad del software, adecuado para proyectos donde la calidad es crítica.	Poco flexible, similar al modelo en cascada.

Metodologías de desarrollo de software

Agile:

- **Descripción:** Un enfoque iterativo e incremental que promueve la flexibilidad y la colaboración.
- **Principios:** Entregas rápidas, colaboración estrecha con el cliente, adaptación a cambios.
- **Métodos Populares:** Scrum, Kanban.
- **Ventajas:** Alta flexibilidad, enfoque en la satisfacción del cliente.
- **Desventajas:** Puede ser difícil escalar para proyectos grandes.

Scrum:

- **Descripción:** Un marco ágil que utiliza sprints cortos y reuniones diarias (scrums) para gestionar el desarrollo.
- **Roles:** Product Owner, Scrum Master, Development Team.
- **Ventajas:** Alta transparencia, mejora continua, entrega rápida.
- **Desventajas:** Requiere disciplina, puede ser difícil de implementar correctamente.

Kanban:

- **Descripción:** Un método ágil que utiliza un tablero visual para gestionar el flujo de trabajo.
- **Principios:** Visualización del trabajo, limitación del trabajo en curso, gestión del flujo.
- **Ventajas:** Enfoque en la mejora continua, flexibilidad.
- **Desventajas:** Puede ser menos estructurado que otros métodos ágiles.

Extreme Programming (XP):

- **Descripción:** Un enfoque ágil centrado en la mejora de la calidad del software y la capacidad de respuesta a los cambios.
- **Prácticas:** Desarrollo guiado por pruebas (TDD), programación en pareja, integración continua.
- **Ventajas:** Alta calidad de software, adaptabilidad.
- **Desventajas:** Requiere un alto nivel de disciplina y colaboración.

Principios fundamentales de la Programación orientada a objetos y su aplicación en sistema:

1. **Encapsulamiento:**
    - **Descripción:** Consiste en agrupar datos y métodos que operan sobre esos datos en una sola entidad llamada "objeto".
    - **Ventaja:** Reduce la complejidad y mejora la mantenibilidad del código. Los detalles internos de un objeto están ocultos y solo se expone una interfaz pública.
    - **Aplicación:** En sistemas complejos, el encapsulamiento permite modularizar el código, facilitando su mantenimiento y extensión.
2. **Abstracción:**
    - **Descripción:** Permite definir el comportamiento y las propiedades esenciales de un objeto, ignorando los detalles no esenciales.
    - **Ventaja:** Simplifica el diseño y permite enfocarse en lo que realmente importa.
    - **Aplicación:** Facilita la creación de modelos de sistemas al enfocarse en las características importantes de los objetos.
3. **Herencia:**
    - **Descripción:** Permite que una clase (subclase) hereda propiedades y métodos de otra clase (superclase).
    - **Ventaja:** Reutiliza código y facilita la creación de jerarquías de clases.
    - **Aplicación:** En el diseño de sistemas, la herencia permite crear clases especializadas a partir de clases generales, lo que facilita la expansión del sistema.
4. **Polimorfismo:**
    - **Descripción:** Permite que un solo método o función tenga diferentes comportamientos según el objeto que lo invoque.
    - **Ventaja:** Mejora la flexibilidad y escalabilidad del código.
    - **Aplicación:** En sistemas, el polimorfismo permite que diferentes objetos sean tratados de manera uniforme, facilitando la implementación de funcionalidades extensibles.

Aplicación en Sistemas

La programación orientada a objetos se utiliza en el diseño de sistemas para crear software modular, reutilizable y fácil de mantener. Aquí te dejo algunos ejemplos de su aplicación:

1. **Desarrollo de Interfaces de Usuario (UI):**

    - Las interfaces gráficas se diseñan utilizando objetos que representan elementos de la UI, como botones, cuadros de texto y ventanas.
2. **Sistemas de Gestión de Bases de Datos:**
    - Los objetos representan entidades de la base de datos, permitiendo manipular los datos de manera más intuitiva y eficiente.
3. **Desarrollo de Videojuegos:**
    - Los personajes, escenarios y objetos del juego se representan como objetos, lo que facilita la creación y gestión del contenido del juego.
4. **Aplicaciones Web:**
    - Las aplicaciones web modernas utilizan marcos de trabajo orientados a objetos, como Django (Python) o Ruby on Rails (Ruby), para estructurar el código y manejar solicitudes HTTP.

Arquitecturas de software, y su impacto en escalabilidad y eficiencia:

1. **Monolítica:**
    - **Descripción:** Toda la funcionalidad del software se agrupa en una sola aplicación unificada.
    - **Impacto en Escalabilidad:** Escalar una aplicación monolítica puede ser desafiante, ya que cualquier cambio o aumento en la carga afecta a toda la aplicación.
    - **Impacto en Eficiencia:** Puede ser eficiente en sistemas pequeños, pero su rendimiento puede deteriorarse a medida que la aplicación crece.
2. **Cliente-Servidor:**
    - **Descripción:** La funcionalidad se divide entre clientes que solicitan servicios y servidores que los proporcionan.
    - **Impacto en Escalabilidad:** Permite escalar horizontalmente añadiendo más servidores para manejar mayores cargas de trabajo.
    - **Impacto en Eficiencia:** Mejora la eficiencia distribuyendo tareas entre clientes y servidores, pero puede enfrentar problemas de latencia y sobrecarga del servidor.
3. **Microservicios:**
    - **Descripción:** El software se divide en pequeños servicios independientes que pueden desplegarse y escalar por separado.
    - **Impacto en Escalabilidad:** Facilita la escalabilidad horizontal, ya que cada microservicio se puede escalar de forma independiente según sea necesario.
    - **Impacto en Eficiencia:** Aumenta la eficiencia operativa y permite una mejor utilización de los recursos, pero puede introducir complejidad en la gestión y comunicación entre servicios.

4. **Arquitectura en N Capas (N-tier):**
    - **Descripción:** Divide la aplicación en capas lógicas, como la capa de presentación, lógica de negocio y acceso a datos.
    - **Impacto en Escalabilidad:** Facilita la escalabilidad al permitir que cada capa se escale de manera independiente.
    - **Impacto en Eficiencia:** Mejora la eficiencia al separar responsabilidades y reducir la carga en cada capa, aunque puede incrementar la latencia debido a la comunicación entre capas.
5. **Arquitectura Orientada a Servicios (SOA):**
    - **Descripción:** Similar a los microservicios, pero se enfoca en proporcionar servicios reutilizables y desacoplados que pueden ser consumidos por diferentes aplicaciones.
    - **Impacto en Escalabilidad:** Facilita la escalabilidad al permitir que los servicios se desplieguen y escalan de manera independiente.
    - **Impacto en Eficiencia:** Mejora la eficiencia al permitir la reutilización de servicios, pero puede introducir complejidad en la gestión de servicios y la orquestación.

Impacto en Escalabilidad y Eficiencia

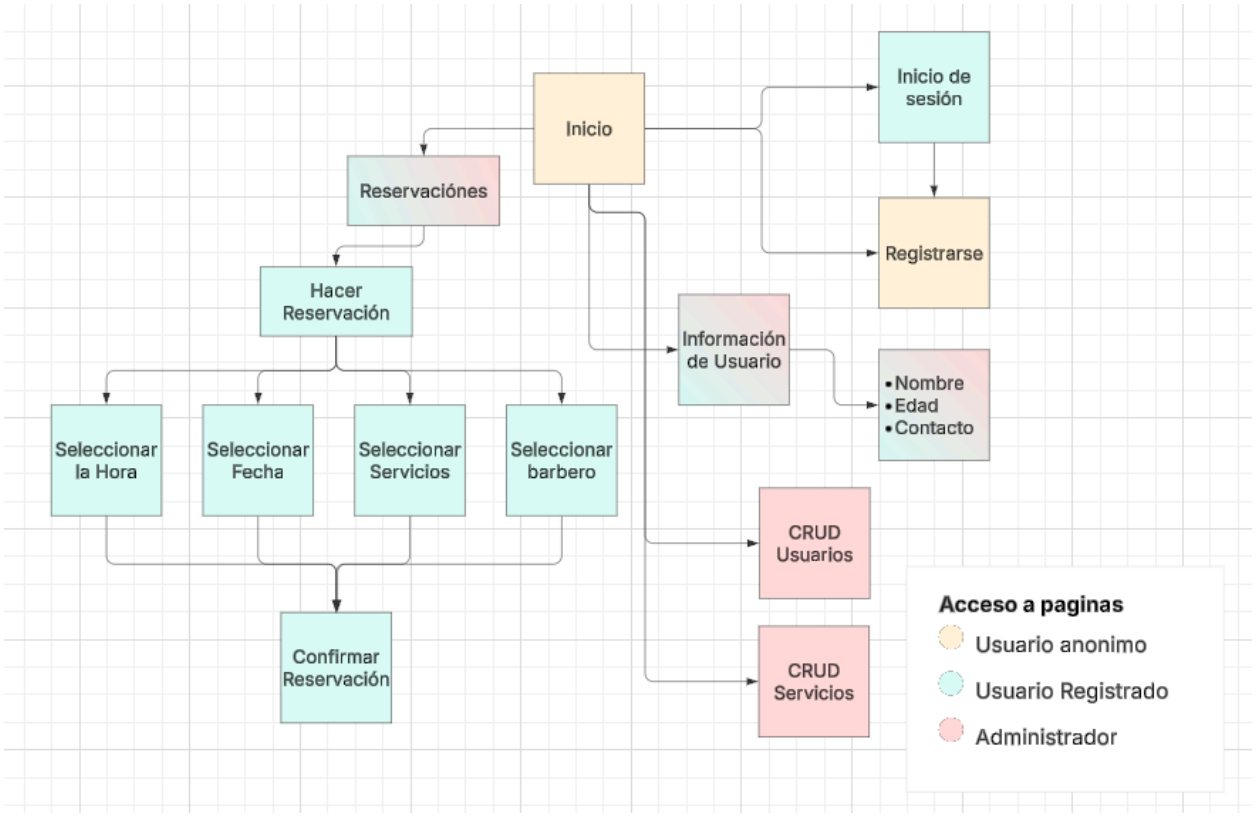
- **Escalabilidad:**
    - La capacidad de un sistema para manejar un aumento en la carga de trabajo es fundamental para sistemas que requieren crecimiento. Arquitecturas como los microservicios y la arquitectura en N capas permiten una mayor flexibilidad en la escalabilidad horizontal, mientras que las aplicaciones monolíticas y cliente-servidor pueden enfrentar limitaciones en este aspecto.
- **Eficiencia:**
    - La eficiencia en el uso de recursos y la capacidad de respuesta del sistema son cruciales para un rendimiento óptimo. Arquitecturas que separan responsabilidades y distribuyen la carga, como los microservicios y la arquitectura en N capas, tienden a ser más eficientes, aunque pueden requerir una mayor gestión y coordinación. Las arquitecturas monolíticas, por otro lado, pueden ser eficientes en sistemas pequeños, pero pueden enfrentar problemas de rendimiento a medida que crecen.

2. Aplicación Práctica de los Contenidos Conceptuales en la Resolución de un Caso

Descripción del problema que se busca resolver con nuestro proyecto:

En este proyecto buscamos desarrollar una página web para un negocio de barbería pequeña, el enfoque de esta página web es el proporcionar un servicio a la barbería para que los clientes puedan hacer citas de manera rápida a través de la página. El cliente podrá seleccionar la hora, el día y qué servicios desea.

Diagrama de flujo del funcionamiento del sistema:



Requisitos funcionales y no funcionales:

- Funcionales:
    - Cliente: Identificarse, Seleccionar fecha, hora, servicio y barbero
    - Empleado: Poder ver las reservas, fechas, horas, servicio, y la información del cliente
    - Administrador: Poder ver y eliminar reservas hechas, además de ver los usuarios, poder eliminar usuarios, agregar y eliminar servicios además de modificar los precios de cada uno.

- No funcionales:
    - Ser intuitivo para el cliente
    - Disponibilidad 24/7
    - Compatible para todos los dispositivos
    - Confirmar identidad del cliente

Codificación de los módulos de prueba para validar la arquitectura (fragmentos de código):

```
1  create database Barberia;
2  use Barberia;
3
4  create table clientes (
5      id_c int not null auto_increment,
6      nombre varchar(50) not null,
7      email varchar(50) not null,
8      telefono int not null,
9      edad int not null,
10     rol int not null,
11     primary key(id_c),
12     foreign key(rol) references roles(rol)
13 );
14
15 create table servicios (
16     id_s int not null auto_increment,
17     descr varchar(100),
18     precio float not null,
19     primary key(id_s)
20 );
```

```sql
create table reserv (
    num_res int not null,
    hora datetime not null,
    fecha date not null,
    id_s int not null,
    id_c int not null,
    primary key(num_res),
    foreign key(id_s) references servicios(id_s),
    foreign key(id_c) references clientes(id_c)
);
```

```sql
create table roles (
    rol int not null auto_increment,
    privilegios varchar(100) not null,
    primary key(rol)
);
```

```sql
create table admin (
    id_adm int not null auto_increment,
    nombre varchar(10) not null,
    rol int not null,
    primary key(id_adm),
    foreign key(rol) references roles(rol)
);
```

```sql
create table barberos (
    id_bar int not null auto_increment,
    nombre varchar(50) not null,
    disp boolean not null,
    primary key(id_bar)
);
```

```sql
alter table reserv add column id_bar int not null, add foreign key(id_bar) references barberos(id_bar);
```

3. Claridad Declarativa en la Explicación de los Procedimientos

Metodología utilizada:

Dentro de las metodologías ágiles, creemos que para este proyecto seria mas optimo utilizar Scrum, ya que es una de las más populares y adecuadas para este tipo de proyectos. Esto debido a la flexibilidad de proporciona, el hecho de que el proyecto se divide en pequeñas partes y permite a el equipo tener una buena comunicación.

Requerimientos del sistema (análisis de necesidades):

- Diseño intuitivo: La página debe ser fácil de navegar, con un proceso de reserva claro y sencillo.
- Calendario de disponibilidad: Muestra los horarios disponibles para reservas, teniendo en cuenta los días y horas de operación de la barbería.
- Selección de servicios: Permite a los clientes elegir entre los servicios.
- Duración de los servicios: Se debe de tener en cuenta la duración de cada servicio para evitar problemas de coordinación en las reservas.
- Confirmación de reserva: Envía una confirmación automática por correo electrónico o SMS con los detalles de la reserva.
- Cancelaciones y modificaciones: Permite a los clientes cancelar o modificar sus reservas fácilmente.
- Capacitación del personal: El personal deberá estar capacitado para usar el sistema de manera adecuada.

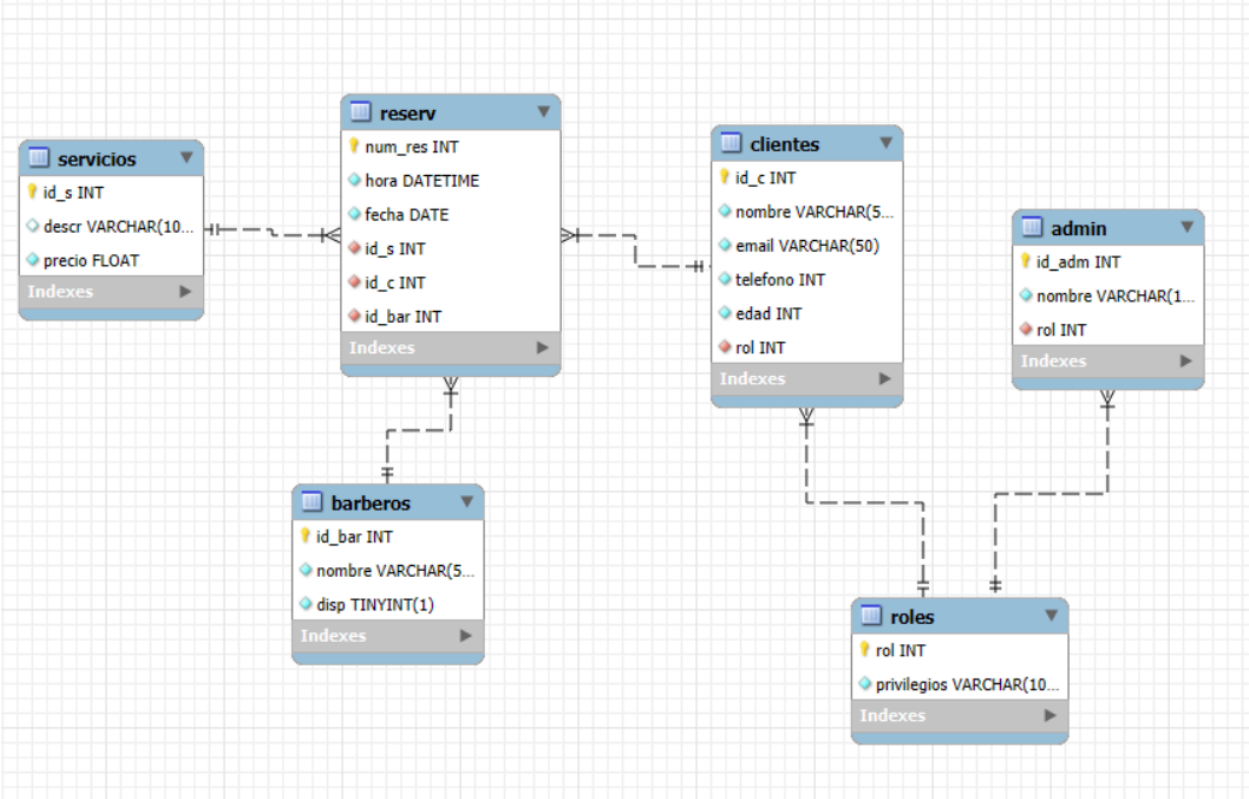
Diseño de la estructura modular del sistema:

1. Frontend
    - Componentes: Componentes reutilizables como Header, Footer y Formulario de Reserva.
    - Páginas: Páginas principales como Inicio, Servicios, Reservas y Confirmación.
    - Styles: Archivos de estilos globales y específicos para cada página.

2. Backend
    - Modelos: Modelos de base de datos para clientes, servicios, barberos y reservas.
    - Procesos: Procesamiento de entrada y salidas de datos

3. Base de Datos
    - Migraciones: Archivos para crear y modificar tablas en la base de datos.
    - Datos: Datos de prueba para inicializar la base de datos.

Diagramas ENTIDAD-RELACIÓN:



Implementación inicial de componentes clave del sistema:

1. Frontend:

- Página de Inicio: Descripción breve de la barbería y botón de "Reservar ahora".
- Formulario de reserva: Selección de fecha, hora, servicio y datos del cliente.
- Página de Confirmación: Muestra detalles de la reserva y opciones para cancelar o modificar.

2. Backend:
- Base de Datos: Tablas para clientes, servicios y reservas.
- API: Endpoints para obtener servicios, crear reservas y gestionar modificaciones.
- Lógica de Reservas: Verificación de disponibilidad y confirmación automática.

3. Notificaciones
- Confirmación de reserva: Envío automático de correo o SMS con los detalles.
- Recordatorios: Envío de recordatorio un día antes o unas horas antes de la cita.

Evaluación preliminar del rendimiento del sistema en un entorno de prueba:

1. Frontend
   - Probar que el formulario de reservas funcione correctamente.
   - Verificar que las páginas carguen rápido, especialmente en móviles.
   - Asegurar que el calendario interactivo no tenga retrasos.

2. Backend
   - Probar que los endpoints respondan rápido.
   - Verificar que la base de datos maneje bien las reservas simultáneas.
   - Asegurar que las transacciones (crear, modificar, cancelar reservas) no fallen.

3. Notificaciones
   - Confirmar que los correos y SMS de reserva se envíen al instante.
   - Verificar que los recordatorios lleguen a tiempo (un día o horas antes).
   - Probar que funcione con diferentes proveedores de correo y SMS.

4. Objetividad en la Aplicación de los Conceptos

Relación entre los objetivos del proyecto y los resultados obtenidos hasta el momento:

Dado que el objetivo principal era desarrollar una plataforma funcional para que los clientes de la barbería pudieran agendar citas de manera rápida y sencilla, la medición del éxito del proyecto puede evaluarse en función de los siguientes aspectos:

1. **Usabilidad y experiencia del usuario**: Si los clientes encuentran la plataforma intuitiva y fácil de usar para reservar citas, significa que se logró el objetivo de proporcionar un servicio eficiente. En nuestro caso teniendo una clara visión del resultado final de nuestro proyecto, sabemos que el cliente se sentirá satisfecho con el funcionamiento de la página.
2. **Eficiencia en la gestión de citas**: Si la barbería puede administrar mejor sus reservas y reducir tiempos de espera o confusiones en la agenda, entonces la página ha cumplido con su propósito.
3. **Tasa de conversión y uso de la plataforma**: Si un alto porcentaje de clientes utilizan la página web para agendar sus citas en lugar de métodos tradicionales (llamadas, visitas en persona), esto indica que la solución es efectiva y valorada. Por lo mismo, adecuamos un sistema que sea capaz de digerir altas cantidades de clientes y que a su vez sea sencillo de utilizar.

4. **Satisfacción del usuario**: Comentarios positivos de los clientes sobre la facilidad de uso y la conveniencia del sistema reflejan que la página ha mejorado su experiencia y, por lo tanto, ha contribuido a la satisfacción de su estadía en la plataforma.

Justificación de las decisiones tomadas en el diseño e implementación:

Nosotros decidimos darle el aspecto ya mencionado dado su fácil uso en las distintas pruebas que le hemos realizado, por otra parte, nos aseguramos de que nuestra página cumpla con lo que cualquier cliente desea encontrarse de un lugar de calidad.
Otro punto importante a considerar es que el diseño hace de nuestro proyecto algo atractivo visualmente y sencillo de manipular para que ningún futuro cliente tenga dificultades a la hora de manejar nuestro sistema.

Áreas de mejora en cuanto a la optimización del código y la experiencia del usuario:

Nuestras posibles áreas de mejora en nuestro código serían las siguientes:

- Mejoras en la funcionalidad: Esto suponiendo que al finalizar el código, no cuente con ciertas funciones que no hagan de la experiencia del usuario algo grato para él. Como por ejemplo, confirmaciones y recordatorios automáticos, evitar reservas dobles verificando en tiempo real la disponibilidad de los barberos, etc.
- Seguridad: Se podría indagar un poco más a fondo para así llegar al punto en donde necesitemos de cierto protocolo de seguridad para así dar un mejor servicio y proteger nuestros datos.

5. Dificultades Encontradas y Próximos Pasos

Retos y problemas detectados: (Errores, ajustes en el diseño, cambios de enfoque, etc.)

Después de realizar las tablas se encontraron algunos errores como:
Falta de normalización en la tabla reserv:

- La tabla reserv usa columnas separadas para fecha y hora. Sería más eficiente usar una sola columna de tipo datetime.
- No hay una columna para el estado de la reserva (por ejemplo, "pendiente", "confirmada", "cancelada").
- La columna edad no es necesaria y podría generar problemas de privacidad.

- La columna teléfono está definida como int, lo que no es adecuado para números de teléfono (debería ser varchar).
- No hay una tabla intermedia que defina qué servicios ofrece cada barbero.
- No hay una relación directa entre admin y otras tablas.
- No hay índices en columnas como fecha y hora en la tabla reserv, lo que podría ralentizar las consultas.

Acciones a seguir: (Mejoras a implementar en la siguiente fase del proyecto.)

- Normalizar la tabla reserv combinando fecha y hora.
- Eliminar o corregir columnas innecesarias o mal diseñadas (edad, teléfono).
- Crear una tabla intermedia para relacionar barberos y servicios.
- Mejorar la tabla admin y agregar índices para optimizar consultas.
- Agregar restricciones de validación para evitar datos inconsistentes.

Fecha estimada de finalización de la siguiente fase: [Día/Mes/Año]

En base a los tiempos del semestre se podría estimar la siguiente fase para el 29/02/2025

Comentarios Adicionales:

De momento hemos iniciado en lo más básico, pero durante estos días nos podremos enfocar en desarrollar la página, anotar y resolver cualquier problema que se pueda llegar a dar, consideramos que para que este proyecto sea exitoso deberá de haber una buena comunicación entre todos los integrantes del proyecto.

Referencias

Iammalf. (2024, October 21). Los 4 principios fundamentales de la programación orientada a Objetos POO. *Diseño De Paginas Web En Cusco*. https://webdesigncusco.com/los-4-principios-fundamentales-de-la-programacion-orientada-a-objetos-poo/

Introducción. (2023, June 1). Portal Académico Del CCH. https://portalacademico.cch.unam.mx/cibernetica2/principios-programacion-orientada-a-objetos/introduccion

SLU, I. (2025, February 18). *Conceptos básicos de la programación orientada a objetos POO - STUCOM Centro d'Estudios*. STUCOM Centro D'Estudios. https://stucom.com/es/blog/conceptos-basicos-de-la-programacion-orientada-a-objetos-poo/

Garcia, C. (2023, August 30). *Escalabilidad y rendimiento: los pilares de una arquitectura de software exitosa*. AppMaster - Ultimate All-in No-code Platform. https://appmaster.io/es/blog/arquitectura-de-software-de-escalabilidad-y-rendimiento

MasonCoding. (2023, October 26). Seguridad, Escalabilidad y Rendimiento en la Arquitectura de Software. *Medium*. https://medium.com/@Masoncoding/seguridad-escalabilidad-y-rendimiento-en-la-arquitectura-de-software-7e9669b28873

Saverin. (2024, July 25). *Arquitectura de Software para Proyectos Robustos: Escalabilidad, Rendimiento y Sostenibilidad*. Saverin. https://saverin.solutions/arquitectura-de-software-para-proyectos-robustos-escalabilidad-rendimiento-y-sostenibilidad/

Admin. (2023, February 5). *Cuadro comparativo de metodologías de software.* Cuadros Comparativos. https://cuadrocomparativode.net/cuadro-comparativo-de-metodologias-de-software/

Antoniolidia. (2023, June 17). *Metodologías de desarrollo de software: más usadas y características*. Todo Ingenierías. https://todoingenierias.com/metodologias-de-desarrollo-de-software-mas-usadas-y-caracteristicas/

Inába, S. (2023, April 9). *Programación: Definición, características e importancia*. Soluciones Inába. https://www.inabaweb.com/programacion-definicion-caracteristicas-e-importancia/

Admin. (n.d.). *Que es un sistema en programación*. Programación Desde Cero. https://programacion.top/conceptos/sistema/

Miñan, M. (2024, July 10). *Definición de Sistema en Programación Según autores, Ejemplos y Concepto*. DefinicionWiki. https://definicionwiki.com/definicion-de-sistema-en-programacion-segun-autores-ejemplos-concepto/