**BarberConnect**

**(SAD) Software Architecture Document**

**Versión 1.0**

**Integrante:**

**Identificación de Documento**

|  |  |
| --- | --- |
| **Identificación** | Documento\_01 |
| **Proyecto** | Software para agendamiento. |
| **Versión** | 1.0 |

|  |  |
| --- | --- |
| **Documento mantenido por** | Angel Alejandro Oscategui Pineda |
| **Fecha de última revisión** |  |
| **Fecha de próxima revisión** |  |

|  |  |
| --- | --- |
| **Documento aprobado por** |  |
| **Fecha de última aprobación** |  |

**Historia de cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 01 de Octubre,2024 | 1.0 | instauración de la Arquitectura Lógica | Damián Salazar |
| 02 de Octubre, 2024 | 0.2 | instauración de la Arquitectura de Procesos | Damián Salazar |
| 02 de Octubre, 2024 | 0.1 | instauración de la Arquitectura de Desarrollo | Ángel Oscategui |
| 03 de Octubre, 2024 | 0.3 | Instauración de la Arquitectura Física | Carlos |
| 04 de Octubre, 2024 | 0.1 | Precisión de los Escenarios | Carlos |

**Tabla de Contenidos**

Contenido

[Introducción 3](#_Toc180369411)

[Alcance 4](#_Toc180369412)

[Referencias 5](#_Toc180369413)

[Arquitectura de Software 5](#_Toc180369414)

[Objetivos y Restricciones de la Arquitectura 8](#_Toc180369415)

[Arquitectura Lógica 10](#_Toc180369416)

[Arquitectura de Procesos 11](#_Toc180369417)

[Arquitectura de desarrollo 13](#_Toc180369418)

[Arquitectura física 14](#_Toc180369419)

[Escenarios 15](#_Toc180369420)

[Tamaño y desempeño 21](#_Toc180369421)

[11. Identificado Riesgos 22](#_Toc180369422)

[13. Modelo de datos 23](#_Toc180369423)

[12. Patrones de Diseño 24](#_Toc180369424)

[1.PATRON DECORATOR: 24](#_Toc180369425)

[2.PATRON BUILDER: 27](#_Toc180369426)

[3.PATRON SINGLETON: 28](#_Toc180369427)

[Fase 1: 30](#_Toc180369428)

[1.PRESENTACION DE ATAM: 30](#_Toc180369429)

[2.PRESENTACION DE LOS OBJETIVOS DE NEGOCIOS: 30](#_Toc180369430)

[3.PRESENTACION DE LA ARQUITECTURA: 31](#_Toc180369431)

[FASE 2: 33](#_Toc180369432)

[4.IDENTIFICACION DE PATRONES ARQUITECTONICOS: 33](#_Toc180369433)

[5.GENERACION DEL ARBOL DE UTILIDAD: 34](#_Toc180369434)

[6.ANALIZAR LOS ENFOQUES ARQUITECTONICOS 35](#_Toc180369435)

[FASE3: 36](#_Toc180369436)

[7.PRIORIZACION DE ESCENARIOS 36](#_Toc180369437)

[8.ANALISS DE ENFOQUES: 37](#_Toc180369438)

[FASE4: 38](#_Toc180369439)

[9.PRESENTACION DE RESULTADOS: 38](#_Toc180369440)

[14. CONCLUCIONES*:* 39](#_Toc180369441)

**Índice de figuras**

# Introducción

En el contexto actual, la tecnología ha transformado la manera en que las empresas y usuarios interactúan. La digitalización de servicios es una tendencia en constante crecimiento, y las pequeñas y medianas empresas, como las barberías, no están exentas de este fenómeno. Es en este escenario que nace *BarberConnect*, una plataforma diseñada para facilitar la gestión de citas entre barberos y clientes de manera eficiente y accesible.

*BarberConnect* es una solución web que permite a los usuarios reservar citas en barberías cercanas, optimizando el proceso tradicional de agendamiento y adaptándolo a las necesidades actuales de los consumidores. Esta plataforma no solo ofrece la posibilidad de gestionar citas de manera rápida, sino que también proporciona a los barberos una herramienta para administrar su disponibilidad y horarios de forma sencilla, mejorando así la experiencia de ambos lados.

El desarrollo de *BarberConnect* ha sido una oportunidad para aplicar conocimientos teóricos adquiridos a lo largo de la carrera de **Ingeniería Informática** en un entorno práctico. El proyecto ha requerido investigar y utilizar tecnologías actuales como **Django** para el backend, **React** para el frontend, y **MySQL** para la gestión de bases de datos, asegurando que el sistema sea escalable, seguro y fácil de mantener.

Este informe se estructura en varias secciones donde se detalla el análisis, diseño y arquitectura del sistema, además de las decisiones tecnológicas que guiaron el desarrollo del proyecto. A través de esta plataforma, se espera que *BarberConnect* se convierta en una solución de referencia en la industria, ofreciendo un servicio de calidad a los usuarios y facilitando la gestión diaria de las barberías.

# Alcance

El proyecto *BarberConnect* es una página de agendamiento para barberías que permitirá a los clientes gestionar citas de forma fácil y rápida, optimizando la interacción entre usuarios y barberos. La solución incluirá funcionalidades clave como un sistema de reservas, creación y gestión de perfiles tanto para barberos como para clientes, notificaciones automáticas para recordatorios de citas, y una completa administración de horarios. Aunque se prevé una futura expansión de funcionalidades, en esta fase inicial no se incluirán pagos en línea.

La tecnología implementada no solo está enfocada en ofrecer una plataforma eficiente, sino en garantizar su sostenibilidad y capacidad de adaptación a las necesidades futuras. En un entorno donde la tecnología avanza rápidamente, *BarberConnect* busca ser flexible, asegurando una experiencia estable y escalable que permita agregar nuevas funcionalidades en el futuro sin comprometer el sistema.

A pesar de sus ventajas, como la facilidad de uso y la automatización del proceso de agendamiento, el sistema presenta algunas limitaciones. Actualmente, está diseñado para operar a nivel local y atender un volumen controlado de usuarios.

El alcance de *BarberConnect* también establece expectativas claras para el equipo de desarrollo y las partes interesadas. Sirve como un marco de referencia para asegurarse de que el proyecto se mantenga dentro de los objetivos definidos y que los recursos se gestionen de manera eficiente. La arquitectura del sistema está diseñada para ser compatible con dispositivos móviles y navegadores modernos, lo que asegura su accesibilidad para una amplia variedad de usuarios. Además, se garantiza que la plataforma pueda evolucionar de manera orgánica, permitiendo futuras expansiones que respondan a las demandas del mercado.

# Referencias

A continuación, se listan las referencias a otros documentos:

No hemos referenciado hacia otros documentos externo, ya que toda la documentación se encuentra en este informe.

# Arquitectura de Software

La arquitectura del sistema *BarberConnect* está representada siguiendo el enfoque del framework 4+1 y las recomendaciones del proceso unificado. Las vistas incluidas en esta versión del documento son:

* **Vista de Casos de Uso y Escenarios de Calidad**: Describe los casos de uso más significativos, presenta los actores y una descripción de sus casos de uso asociados. De igual forma describe los escenarios de calidad más relevantes para la arquitectura.

**Caso de uso “registro de usuario”:**

Para realizar el registro de usuario en *BarberConnect*, interactúan tres actores:

* **Usuario**: Persona que desea agendar una cita en una barbería.
* **Sistema**: Plataforma que valida los datos y registra al usuario.
* **Google (opcional)**: Permite el registro rápido a través de la cuenta de Google.

El usuario puede registrarse ingresando directamente sus datos en el sistema (nombre, correo electrónico, contraseña y número de teléfono) o utilizar su cuenta de Google. El sistema valida la información ingresada y crea el perfil del usuario en la base de datos. En el caso de utilizar Google, el sistema solicita los permisos necesarios para extraer la información de la cuenta y proceder con el registro

**Caso de uso “Agendar una Cita”:**

Para agendar una cita en una barbería participan dos actores:

* **Usuario**: Persona que busca un servicio en una barbería.
* **Sistema**: Plataforma que gestiona la agenda de la barbería.

El usuario selecciona una barbería de la lista, elige un barbero disponible, y el sistema le muestra las fechas y horarios disponibles. Tras elegir una hora, el sistema valida la disponibilidad en tiempo real y confirma la reserva. El sistema envía una notificación al usuario confirmando la cita y otra a la barbería para registrar la solicitud en su agenda.

**Escenario de calidad “Usabilidad”:** En este escenario se evalúa la facilidad de uso de la plataforma *BarberConnect*. Se implementa una interfaz sencilla e intuitiva, donde los usuarios pueden realizar sus reservas con pocos clics. Las pruebas realizadas con usuarios nuevos mostraron que podían interactuar con el sistema y agendar una cita sin dificultades ni capacitación previa.

**Escenario de calidad “portabilidad”:** El escenario de calidad basado en la portabilidad verifica la compatibilidad de *BarberConnect* en distintas plataformas y dispositivos. Las pruebas mostraron que el sistema funcionaba correctamente en navegadores móviles y de escritorio, sin afectar la funcionalidad o experiencia de usuario. Esto asegura que los usuarios puedan agendar citas desde cualquier dispositivo con acceso a internet.

* **Vista de Metas y Restricciones**: Describe las restricciones tecnológicas y normativas que influyen en las decisiones arquitectónicas del sistema:

 **Tecnológicas**: El sistema debe ser compatible con dispositivos móviles y de escritorio, por lo que se utilizó un diseño responsive en el frontend (desarrollado en React). La base de datos debe ser escalable y segura, optando por MySQL con optimización de consultas.

 **Restricciones Normativas**: Se cumplen con las normativas de protección de datos personales (GDPR/LPDP) al almacenar la información de los usuarios y barberos. Todo el tráfico se asegura mediante HTTPS.

* **Vista Lógica**: Describe la arquitectura lógica del sistema, presentando los módulos principales y sus responsabilidades. Esta vista usa el tipo de vista "Módulos" para representar la estructura lógica y "Componentes y Conectores" para describir el comportamiento del sistema.

 **Módulo de Usuarios**: Gestiona el registro, autenticación y perfil de los usuarios.

 **Módulo de Barberías**: Maneja la información de cada barbería, incluyendo su disponibilidad y servicios ofrecidos.

 **Módulo de Reservas**: Coordina las citas entre usuarios y barberías, validando la disponibilidad y confirmando las reservas.

 **Módulo de Notificaciones**: Gestiona el envío de notificaciones automáticas por correo electrónico o SMS sobre las reservas realizadas y recordatorios.

* **Vista de Procesos**: Describe los procesos involucrados para darle sentido a la ejecución del sistema, así como sus relaciones de comunicación y sincronización.

 **Proceso de Agendamiento de Citas**: Cuando un usuario selecciona una barbería, elige un barbero disponible y selecciona un horario, el sistema verifica en tiempo real la disponibilidad de ese barbero en el horario seleccionado. Si el slot está disponible, el sistema confirma la reserva y envía notificaciones automáticas tanto al usuario como a la barbería. El proceso se sincroniza con la base de datos para evitar overbooking y asegurar que no haya conflictos de horarios entre diferentes barberos.

 **Proceso de Notificaciones**: Se ejecuta en segundo plano y asegura que tanto los usuarios como las barberías reciban notificaciones sobre la reserva o recordatorios previos a la cita. Este proceso está sincronizado con el sistema de reservas para garantizar la exactitud de la información, y también incluye detalles específicos del barbero seleccionado para cada cita.

* **Vista de Implementación**: Describe los componentes de deployment construidos y sus dependencias.

 **Backend**: Implementado en **Python** usando el framework **Django**, alojado en un **servidor de hosting** que nos proporciona el espacio para alojar la página web. Este componente se comunica con la base de datos **MySQL** para gestionar la información de usuarios, barberías y reservas.

 **Frontend**: Desarrollado en **JavaScript** usando **React**, implementado como una aplicación web responsive. Esto asegura que la plataforma funcione correctamente en cualquier dispositivo, adaptándose a pantallas móviles, tabletas y computadoras.

 **Base de Datos**: **MySQL**, también alojada en el **servidor de hosting**, gestiona todas las entidades del sistema, como usuarios, barberías, horarios y reservas. La base de datos está integrada directamente con el backend para asegurar que toda la información esté sincronizada y disponible en tiempo real.

# Objetivos y Restricciones de la Arquitectura

A continuación, se revisan las metas y restricciones de la arquitectura de software para *BarberConnect*, una página web de agendamiento diseñada para facilitar la conexión entre los clientes y las barberías, permitiendo reservar citas de manera rápida y eficiente.

**Objetivos de la arquitectura**

*BarberConnect* tiene como objetivo principal ofrecer una plataforma ágil y confiable que permita a los usuarios gestionar sus citas en barberías, asegurando una experiencia fluida tanto para clientes como para barberos. El sistema busca optimizar la comunicación, aumentar la eficiencia en la gestión de citas y proporcionar una interfaz accesible y fácil de usar.

* **Desempeño**:

El desempeño es crucial para asegurar la satisfacción de los usuarios. En el caso de *BarberConnect*, el sistema debe responder de manera rápida y eficiente a las solicitudes de reserva, especialmente en un entorno con múltiples usuarios simultáneos.

Para lograr esto, utilizamos una base de datos relacional (MySQL) optimizada con índices que garantizan un acceso rápido a la información. La aplicación está diseñada para minimizar los tiempos de carga, especialmente en la búsqueda de barberías y en la validación de la disponibilidad de horarios.

* **Tolerancia a fallos**:

Es fundamental que *BarberConnect* continúe operando incluso en caso de fallos en algunos de sus componentes. El sistema se ha diseñado con redundancia en los puntos críticos, como la base de datos y el servidor de backend, asegurando que, ante un fallo parcial, el sistema pueda seguir funcionando sin interrupciones.

Para garantizar esto, se utiliza un **entorno de hosting** que incluye mecanismos de respaldo adecuados proporcionados por el proveedor de hosting. Esto asegura que cualquier fallo en el servidor no afecte la disponibilidad general de la plataforma, manteniendo la estabilidad y continuidad del servicio.

* **Seguridad**:

La seguridad es un aspecto clave para proteger los datos personales de los usuarios y barberos que interactúan con la plataforma. *BarberConnect* implementa HTTPS para todas las comunicaciones, asegurando que los datos viajen de manera encriptada entre el cliente y el servidor.

Además, utilizamos autenticación basada en **JWT (JSON Web Tokens)** para gestionar el acceso seguro a las funcionalidades del sistema. Se han implementado pruebas de seguridad, como **fuzzing**, para identificar posibles vulnerabilidades que podrían comprometer los datos de los usuarios

* **Modificabilidad/Reusó**:

La arquitectura de *BarberConnect* está diseñada para ser altamente modificable, lo que permite la fácil incorporación de nuevas funcionalidades o ajustes basados en cambios en los requisitos. Este enfoque modular asegura que los componentes individuales del sistema puedan ser reemplazados o actualizados sin necesidad de reestructurar toda la aplicación.

Además, la reutilización de componentes es clave. El código del backend y el frontend se ha desarrollado utilizando patrones modulares, lo que facilita el mantenimiento y las futuras expansiones de la plataforma, como la integración de pagos en línea o la ampliación a nivel regional.

* **Operatividad**:

El sistema debe cumplir con su propósito básico: permitir la gestión de citas de forma eficiente. *BarberConnect* está diseñado para ofrecer una experiencia de usuario fluida, con una interfaz sencilla y accesible para cualquier usuario, independientemente de su nivel de habilidad técnica.

La operatividad se asegura mediante la implementación de pruebas continuas, tanto en el frontend como en el backend, garantizando que el sistema produzca los resultados esperados bajo distintas condiciones de uso. Además, se prioriza la facilidad de uso, con una interfaz clara y bien organizada que minimiza el esfuerzo requerido por el usuario para completar tareas.

**Restricciones de la Arquitectura**

Existen restricciones que han sido levantadas con los stakeholders del proyecto *BarberConnect*, las cuales se presentan a continuación:

* **Tiempo de construcción**: El plazo disponible para la implementación de la arquitectura es limitado, contando con un máximo de 6 semanas para su desarrollo completo según la planificación establecida para el proyecto. Este tiempo restringe el alcance de las funcionalidades a las más esenciales en esta primera versión.
* **Infraestructura**: El sistema se desplegará utilizando un servicio de **hosting tradicional**, lo que proporciona el espacio necesario para alojar los archivos y la base de datos del sistema sin necesidad de una infraestructura en la nube más compleja. Este hosting ofrece soporte para los lenguajes de programación **Python** (backend) y **JavaScript** (frontend)
* **Otros componentes de software**: No se considerará la adquisición de software propietario o con licencias de pago. Todo el software utilizado será de código abierto, incluyendo frameworks como **Django** (Python) para el backend y **React** (JavaScript) para el frontend. Esto permite reducir costos y mantener la flexibilidad del sistema.
* **Base de datos:** Se utilizará una base de datos **relacional** soportada por el servicio de hosting, como **MySQL**, para gestionar la información de usuarios, barberías, horarios y citas. Esta elección se debe a la necesidad de un sistema que maneje datos estructurados con integridad relacional, y es compatible con el entorno de hosting seleccionado
* **Lenguaje de programación**: El backend de *BarberConnect* será desarrollado en **Python**, utilizando el framework **Django** para manejar la lógica de negocio y la comunicación con la base de datos. Para el frontend, se utilizará **JavaScript** con el framework **React**, asegurando una interfaz de usuario interactiva y responsiva. Esta combinación de tecnologías es compatible con el hosting elegido y facilita la escalabilidad futura del sistema.

# Arquitectura Lógica

A continuación, se presenta una vista lógica de la aplicación expresado en dos diagramas, uno de ellos que muestra la parte estructural o estática de la aplicación (módulos), y otra vista que representa la parte dinámica (componentes y conectores).

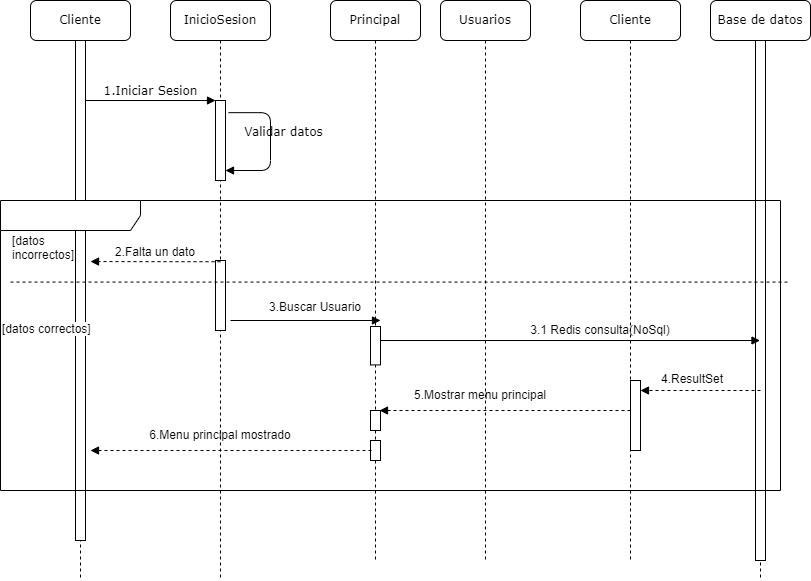
**Parte Estructural**

En el siguiente diagrama de clases se observa que el principal módulo….

***Ilustración 1: Diagrama de clases***

**Parte Dinámica**

La parte dinámica….



***Ilustración 2: Diagrama de secuencia***

# Arquitectura de Procesos

A continuación, se muestra una vista de procesos, en la cual se observa que:

***Ilustración 3: Diagrama de actividad***

# Arquitectura de desarrollo

En esta vista se aprecia que existirán dos módulos principales que contendrán distintas funcionalidades de la aplicación. A continuación, se describen:

**1)A continuación se presenta el diagrama de componentes:**

La relación entre las interfaces, las clases y los datos.

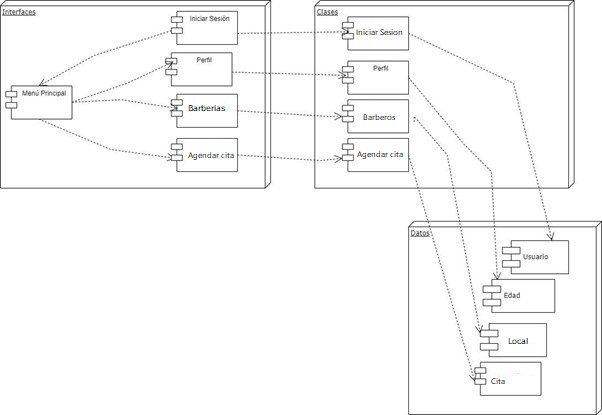
La interfaz de iniciar sesión se comunica con la clase iniciar sesión para que haga la búsqueda

en la tabla de usuarios.

Luego la interfaz llamaría al menú, el menú llamara a las interfaces según donde le damos clic luego.

la interfaz de perfil se comunica con la clase perfil para ver la edad que tengo.

la interfaz de barberías se comunica con la clase barberías para consultar los locales disponibles.

la interfaz de Agendar Cita se comunica con la clase Agendar Cita, para ver las citas disponibles.

***Ilustración 4: Vista de Implementación (Diagramas de componentes o paquetes)***

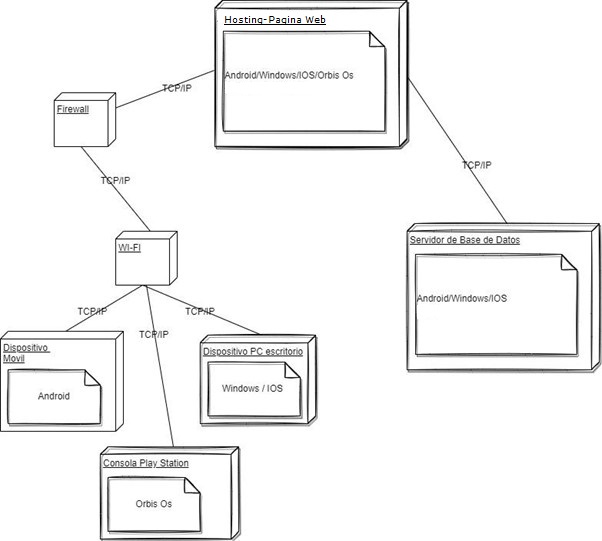
# Arquitectura física

En esta vista se despliegan los nodos que participan con el sistema. Los nodos principales son los nodos Servidor de Integración. Características a continuación:

**1.A continuación vemos el diagrama de despliegue:**

El dispositivo móvil tiene que conectarse al wifi, pasa el muro de seguridad del firewall

y luego ingresa al servidor de la aplicación y a su vez se conectará a la base de datos y este es el concepto de MVC modelo vista controlador.

**

***Ilustración 6: Diagrama de Despliegue***

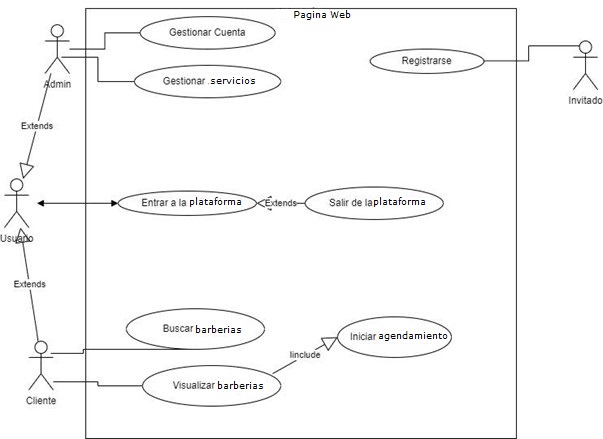
# Escenarios

Los escenarios son útiles para especificar los atributos de calidad, es un requerimiento especifico a una sola categoría de atributo de calidad compuesto por 6 partes, se tiene varios escenarios de atributos de calidad, de acuerdo con que atributo queremos atacar del sistema; estos tienen ciertos pasos que d sección describe en detalle el conjunto de escenarios funcionales y no funcionales que obtuvieron la mayor prioridad en el análisis.

Para esto se presenta y describe el diagrama de casos de uso y los casos de uso prioritarios, así como los escenarios en que uno o más atributos de calidad se ven involucrados de manera significativa.

***Modelo de Casos de Uso***

El modelo de casos de uso puede ser encontrado en el documento “Casos de Uso”.

***1.En este primer caso se aborda las funcionalidades más relevantes de los distintos tipos de usuarios.*****

**Especificación de Casos de Uso Relevantes**

Los casos de uso considerados los más relevantes para el desarrollo de la arquitectura fueron determinados. Los criterios usados para dicha determinación fueron:

* Su implementación implica varios nodos de la vista de despliegue.
* Su implementación es de alto riesgo.
* Incluye muchos conceptos y relaciones del dominio.
* Incluye posibles escenarios críticos de calidad.

**Especificación de los Escenarios de Calidad Relevantes**

Después de un análisis en conjunto con los stakeholders, los escenarios de calidad se expresan a continuación:

Los escenarios están compuestos de 6 pasos y se especifica cada uno de ellos, en los siguientes escenarios de calidad asignados para poder especificar de manera más detalla los atributos de calidad.

Los escenarios seleccionados son los siguientes.

**Escenario de calidad: Portabilidad de la aplicación.**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

|  |  |
| --- | --- |
| Escenario de Calidad N° | |
| Atributo de Calidad Asociado (Característica): Portabilidad.  Subcaracterística: Adaptabilidad. | |
| Descripción: La aplicación de celular debe ser compatible en las distintas plataformas existentes. | |
| Fuente del Estímulo: | Usuario. |
| Estimulo: | Acceder a la aplicación de celular. |
| Artefacto: | Aplicación. |
| Ambiente: | Explotación. |
| Respuesta: | Debe funcionar de manera correcta en cualquier tipo de dispositivo y plataformas de softwares distintas existentes. |
| Medida de Respuesta: | Tiene que ser probado en distintos, dispositivos, navegadores y sistemas operativos. |

**Escenario de calidad: Buscador.**

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**

|  |  |
| --- | --- |
| Escenario de Calidad N° | |
| Atributo de Calidad Asociado (Característica): Eficiencia de desempeño.  Subcaracterística: Comportamiento temporal. | |
| Descripción: El usuario usa el buscador para encontrar barberías. | |
| Fuente del Estímulo: | Usuario. |
| Estimulo: | Realizar una consulta en el buscador. |
| Artefacto: | En todo el sistema. |
| Ambiente: | Aplicación finalizada y operativa. |
| Respuesta: | Muestra el resultado de la consulta. |
| Medida de Respuesta: | Tiempo no mayor a 4 segundos. |

Escenario de calidad:Ingreso fallido a la cuenta.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

|  |  |
| --- | --- |
| Escenario de Calidad N° | |
| Atributo de Calidad Asociado (Característica): Seguridad.  Subcaracterística: Confidencialidad. | |
| Descripción: Un usuario desconocido intento acceder a la cuenta de otra persona con fines maliciosos. | |
| Fuente del Estímulo: | Usuario desconocido. |
| Estimulo: | Introduce el nickname y contraseñas incorrectas. |
| Artefacto: | Operación normal. |
| Ambiente: | Condición Normal. |
| Respuesta: | Registro de intento en la base de datos y bloqueo de la cuenta. |
| Medida de Respuesta: | Después de 4 intentos. |

# Tamaño y desempeño

Las principales decisiones arquitectónicas para *BarberConnect* se tomaron considerando la restricción de **Tiempo de Construcción**. Dado que el proyecto debe completarse en un plazo ajustado de 6 semanas, se priorizó la adopción de una arquitectura conocida y de bajo riesgo en su implementación, utilizando tecnologías como **Django** para el backend y **React** para el frontend. Esto asegura una curva de aprendizaje mínima para el equipo y un desarrollo eficiente.

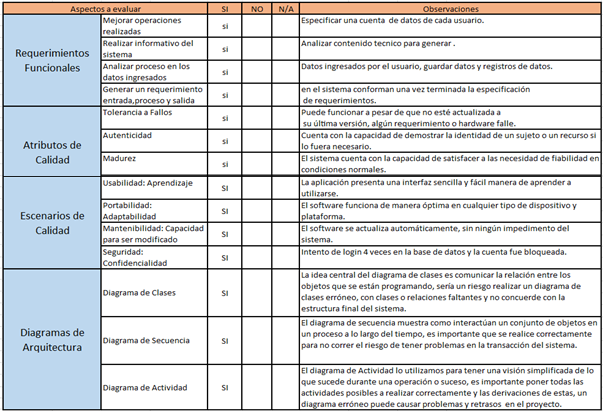
La arquitectura se diseñó de manera modular, con el objetivo principal de separar las responsabilidades. Esta modularización no solo facilita la paralelización del trabajo, permitiendo que los distintos componentes puedan desarrollarse de manera independiente, sino que también habilita el testeo unitario de cada módulo. Al garantizar que cada parte del sistema sea testeable de manera individual, reducimos la tasa de fallos y mejoramos la calidad global del sistema.

Además, la arquitectura se diseñó considerando la **Infraestructura** disponible y las necesidades de **Tolerancia a Fallos**. El sistema utilizará un servicio de hosting que no ofrece replicación ni balanceo de carga en esta fase, lo que nos llevó a estructurar la aplicación de manera que pueda ser escalada en el futuro con servicios adicionales si es necesario.

El escenario de calidad relacionado con la **Mantenibilidad** nos guio hacia un diseño que prioriza la separación de responsabilidades, siguiendo patrones de diseño como el **patrón provider**. Esto permite que *BarberConnect* delegue la obtención de información a sistemas externos a través de piezas de software desacopladas, facilitando su extensibilidad en el futuro. Este enfoque asegura que, en caso de que se necesiten integrar nuevos servicios o modificar los actuales, el sistema pueda adaptarse sin grandes reestructuraciones.

Por último, se eligió una estrategia simple y efectiva para manejar las notificaciones de agendamiento, mediante el envío automático de correos electrónicos cuando se agenda una nueva cita. Esto asegura que tanto los clientes como los barberos reciban notificaciones de manera oportuna sin necesidad de intervención manual. Esta estrategia garantiza que el sistema funcione de manera fluida y minimiza el riesgo de fallos en la comunicación, asegurando una experiencia eficiente para todos los usuarios.

# 11. Identificado Riesgos

**

En los análisis retrospectivos de los proyectos evaluados se identificaron falencias en la gestión de riesgos y esto para los proyectos tecnológicos, específicamente para el desarrollo de software, puede significar problemas y deficiencias en las etapas de planeación y ejecución, así como para dar respuesta a hechos inesperados, que son inherentes a la naturaleza de estos proyectos porque están en entornos de alta incertidumbre.

En línea con esto, según un estudio realizado por el Project Management Instituto (PMI), la mala administración del riesgo es un factor determinante que obstaculiza el logro de los objetivos de un proyecto. Por eso, la gestión de riesgos en proyectos de software cada vez cobra mayor relevancia como mecanismo de anticipación y aprovisionamiento para tener una gestión de proyectos más proactiva que reactiva y de esta forma, poder incrementar la tasa de éxito en todas sus ejecuciones.

# 13. Modelo de datos

# 12. Patrones de Diseño

## 1.PATRON DECORATOR:

**Decorator** es un patrón de diseño estructural que te permite añadir funcionalidades a objetos colocando estos objetos dentro de objetos encapsuladores especiales que contienen estas funcionalidades.

¿Y en la práctica cómo funciona?

Existen cuentas premium la cual se dividen en plus o Gold; Cuando una persona se registra, la plataforma web le pregunta si desea ser usuario premium o no, entonces como ese escenario es opcional, como esto no se va a dar para todos los usuarios, algunos aceptan y otros no, podemos definir una clase normal y una clase decorada con esa característica.

¿Y cómo lo podemos aplicar a nuestro código?

PASO 1:

Creamos la clase usuario con algunos atributos, nombre y créditos, creamos 2 objetos de esa clase.

Texto

Descripción generada automáticamente

PASO2:

Creamos el decorador UsuarioPremiums, el cual agregara una característica al usuario normal, sin necesidad de modificar la clase usuario, se crea la clase usuario plus y usuario Gold, que heredaran del usuario premium.

Texto

Descripción generada automáticamente

PASO 3:

Aquí creamos 2 objetos:

Texto

Descripción generada automáticamente

Y veamos el resultado:

:Texto

Descripción generada automáticamente

En el resultado vemos que José es un usuario normal, pero maría es un usuario pluspremium implementadose correctamente el patrón DECORATOR

## 2.PATRON BUILDER:

**Builder** es un patrón de diseño creacional que nos permite construir objetos complejos paso a paso. El patrón nos permite producir distintos tipos y representaciones de un objeto empleando el mismo código de construcción.

¿Y en la práctica cómo funciona?

¡¡¡Construye!!!, Es el primero que se ejecuta cuando creamos un nuevo objeto o una nueva instancia

de nuestra clase, es el encargado de inicializar nuestros objetos con ciertas características

o reglas que nosotros especifiquemos y menos líneas de código.

## 3.PATRON SINGLETON:

**Singleton** es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia.

¿Y en la práctica cómo funciona?

Tener una única instancia de una clase durante toda aplicación por ende tiene

que ser privado ya que si fuera pública podría crearse más instancia de esa clase,

imaginemos que existe una lista de comunas que mostrar al registrarse, no tendría sentido

devolver a cada usuario que se registre en el sistema la lista de comunas, sería mejor tener

un único objeto el cual sea instanciado una única vez y por cada usuario que necesite esta información

y esto en consumo de memoria va a ser mucho mejor.

¿Y cómo lo podemos aplicar a nuestro código?

PASO 1:

En este código creamos una clase singleton, la cual parece que creamos 5 objetos, pero en realidad es el mismo, y se imprime 5 veces, pero es una sola instancia, y para poder confirmar estos

podemos apoyarnos de la palabra reservada “is”, ya que podemos conocer si un objeto es otro objeto o el mismo.

Texto

Descripción generada automáticamente

PASO 2:

Como se ve el resultado, pareciera que son 5 objetos, pero en realidad es el mismo,

cuando hacemos la comparación en: print (a is b is c is d is e) nos da como resultado “True”.

Texto

Descripción generada automáticamente

13. EVALUACION APLICANDO MODEL ATAM:

# Fase 1:

## 1.PRESENTACION DE ATAM:

En este estudio se establece la evaluación arquitectónica del diseño del software asociado al funcionamiento de nuestra plataforma de agendamiento para barberías, *BarberConnect*. En este paso, se presenta el modelo ATAM a los stakeholders, utilizando este tiempo para explicar el proceso que se seguirá para la evaluación de la arquitectura del sistema. Se fijan las expectativas de las actividades y se contextualiza el modelo con las necesidades del proyecto *BarberConnect*, estableciendo cómo se evaluará la eficiencia, seguridad, y escalabilidad de la arquitectura del sistema.

## 2.PRESENTACION DE LOS OBJETIVOS DE NEGOCIOS:

El líder del proyecto presentó los objetivos de negocio a los stakeholders, tanto desde el punto de vista de los desarrolladores como desde la perspectiva de los usuarios y servicios que se pretende brindar. Para los desarrolladores, el software de *BarberConnect* debe considerar los siguientes conductores del sistema:

* + **Administración de la información** de los usuarios y barberías, asegurando la integridad y disponibilidad de los datos.
  + **Control y auditoría de información** en la base de datos para garantizar la precisión y seguridad.
  + **Depuración de información** en casos de inconsistencias o colisiones en los datos gestionados.
  + **Soporte para los módulos principales del sistema** (reservas, notificaciones, perfil de usuarios/barberos).
  + **Facilidad de creación de nuevos módulos** en función de los requerimientos futuros de los usuarios, como la integración de pagos online.
  + **Manejo de seguridad** a nivel de acceso a datos sensibles de usuarios y barberos, siguiendo los estándares de confidencialidad e integridad.
  + **Dimensionamiento de las capacidades del sistema** para evitar cuellos de botella en el agendamiento masivo de citas.
  + **Capacidad de gestionar múltiples conexiones simultáneas** en el sistema, especialmente durante horas de alta demanda.

Los atributos de calidad de mayor prioridad para la plataforma incluyen:

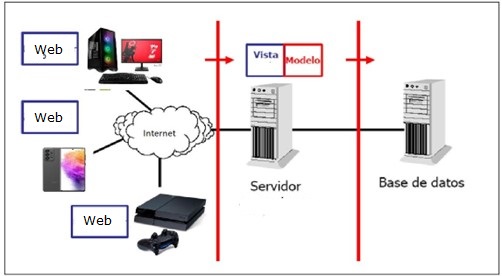
* + **Desempeño**: El sistema debe ofrecer tiempos de respuesta rápidos para la búsqueda de barberías y la confirmación de citas, asegurando una experiencia fluida para los usuarios.
  + **Usabilidad**: La interfaz debe ser fácil de entender y usar por los clientes y barberos, minimizando la curva de aprendizaje y maximizando la eficiencia.
  + **Seguridad**: Dada la importancia de la información personal de los usuarios (como nombres, correos, y preferencias), el sistema debe garantizar la **confidencialidad e integridad** de los datos mediante cifrado y autenticación segura.
  + **Disponibilidad**: El sistema debe estar disponible en todo momento, asegurando que las reservas puedan realizarse sin interrupciones incluso durante horas pico.
  + **Modificabilidad**: La arquitectura de *BarberConnect* debe ser flexible para incorporar nuevos escenarios, entidades o funcionalidades, como la expansión a nuevas regiones o la integración con nuevas pasarelas de pago.
  + **Escalabilidad**: El sistema debe ser lo suficientemente flexible para crecer en capacidad, tanto en términos de usuarios concurrentes como en la posibilidad de añadir nuevas barberías o servicios sin afectar el rendimiento general del sistema.

## 3.PRESENTACION DE LA ARQUITECTURA:

Durante las interacciones que tuvo el equipo de evaluación con los desarrolladores y líderes del proyecto *BarberConnect*, surgieron varias vistas y propuestas arquitectónicas. Los resultados clave se resumen a continuación:

1. **Vistas arquitectónicas:**
   * **Herramientas de desarrollo**: El software está siendo desarrollado utilizando **Django** para el backend, y **HTML**, **CSS** y **React** para el frontend. Esta combinación garantiza un desarrollo ágil, modular y compatible con las necesidades de la plataforma.
   * **Manejo de datos**: Se ha decidido utilizar **MySQL** como motor central de base de datos en el entorno del hosting. MySQL fue elegido por su escalabilidad y robustez para manejar la información estructurada relacionada con usuarios, barberías y citas.
   * **Metodología orientada a objetos**: El sistema sigue un enfoque orientado a objetos, lo que proporciona flexibilidad en la configuración y facilita el mantenimiento del sistema.
   * **Requisitos del navegador**: El sistema es compatible con navegadores modernos como **Google Chrome**, **Firefox**, **Microsoft Edge**, y otros navegadores actuales, asegurando su accesibilidad desde diversos dispositivos y plataformas.
2. **Estilos Arquitectónicos**

* **Arquitectura de 3 capas**: *BarberConnect* sigue una **arquitectura de tres capas**, que se divide en:
  + **Capa de presentación**: Incluye el frontend desarrollado en React, que maneja la interfaz de usuario, y presenta todas las interacciones necesarias para agendar citas y gestionar perfiles de usuarios y barberías.
  + **Capa lógica o de aplicación**: Desarrollada en **Django** (Python), esta capa gestiona la lógica de negocio, como la validación de reservas, la gestión de usuarios, y las notificaciones automáticas.
  + **Capa de datos**: Implementada con **MySQL**, la base de datos almacena toda la información del sistema, como detalles de los usuarios, barberías, citas y servicios ofrecidos.



* **Distribución en nodos**: El sistema se distribuye en tres nodos principales:
* **Cliente (navegador web)**: Los usuarios interactúan con *BarberConnect* a través de un navegador web, enviando solicitudes al servidor web.
* **Servidor Web**: El servidor web, utilizando **Django**, gestiona todas las solicitudes, procesa la lógica de negocio y devuelve respuestas al cliente.
* **Servidor de Base de Datos**: Almacena y gestiona los datos relacionados con los usuarios, barberías y citas. Este servidor es accesado por el servidor web a través de consultas SQL.



* **Estilo de llamada y retorno**: La arquitectura de *BarberConnect* utiliza el estilo de **Llamada y Retorno**, típico en arquitecturas en capas. Cada solicitud del cliente (usuario) genera una llamada al servidor, que procesa la solicitud y devuelve los resultados correspondientes.
* **Procedimientos almacenados**: A nivel de la base de datos, se emplean **procedimientos almacenados** para optimizar las consultas y operaciones comunes como la creación, actualización o eliminación de citas y perfiles. Además, se utilizan **librerías** a nivel de código fuente, que se han depurado a lo largo del desarrollo de la aplicación.
* **Clases y funciones**: Se utilizan clases y funciones dentro del código para invocar los procedimientos almacenados, lo que permite realizar consultas, actualizaciones y eliminaciones de información de manera eficiente a través de las interfaces de usuario.

1. **Requerimientos Arquitectónicos**

Los siguientes **requerimientos arquitectónicos** guían el diseño y desarrollo de *BarberConnect*:

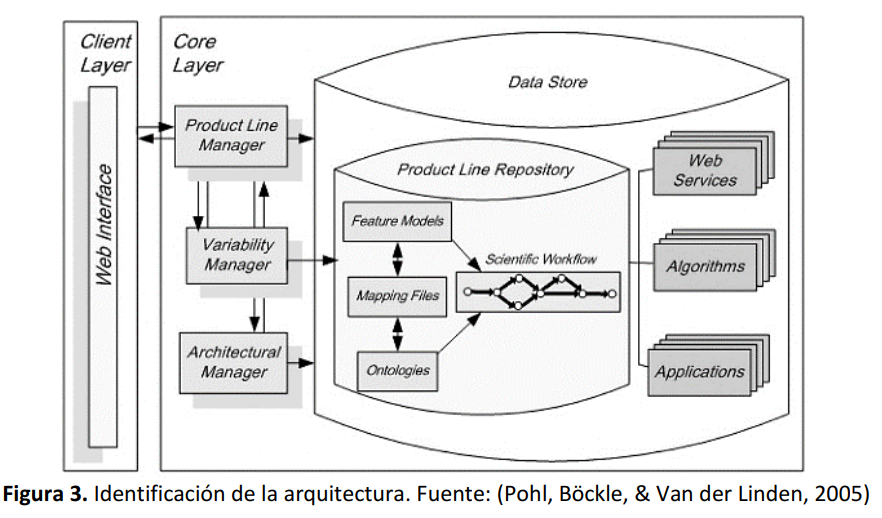
* **Disponibilidad**: El sistema debe ser altamente disponible, asegurando que los usuarios puedan acceder y reservar citas en cualquier momento. En caso de fallos, el sistema debe minimizar el tiempo de inactividad y garantizar que las operaciones críticas, como la gestión de citas, no se vean afectadas.
* **Modificabilidad**: *BarberConnect* debe ser fácilmente modificable para incorporar nuevas funcionalidades y requisitos en el futuro, como la posibilidad de añadir nuevas barberías, servicios adicionales o incluso expandir la plataforma a nivel regional. La estructura orientada a objetos y modular facilita estas modificaciones.
* **Desempeño**: El desempeño es crítico, especialmente durante picos de demanda cuando múltiples usuarios intenten reservar citas al mismo tiempo. El sistema debe manejar estas solicitudes de manera eficiente, proporcionando tiempos de respuesta rápidos para evitar demoras en la confirmación de citas.
* **Seguridad**: La seguridad es un aspecto clave, ya que *BarberConnect* maneja información personal de los usuarios y datos sensibles de las barberías. El sistema debe prevenir accesos no autorizados, garantizando la confidencialidad e integridad de la información a través de medidas de seguridad como la autenticación de usuarios y el cifrado de datos.

# FASE 2:

## 4.IDENTIFICACION DE PATRONES ARQUITECTONICOS:

Después de la presentación de la arquitectura del software de *BarberConnect*, se enumeraron las propuestas arquitectónicas obtenidas a partir de entrevistas con los desarrolladores y líderes del proyecto, así como del análisis de la documentación durante la etapa de preevaluación. Las propuestas arquitectónicas identificadas se enumeran a continuación, aunque no se profundiza en su análisis en esta fase:

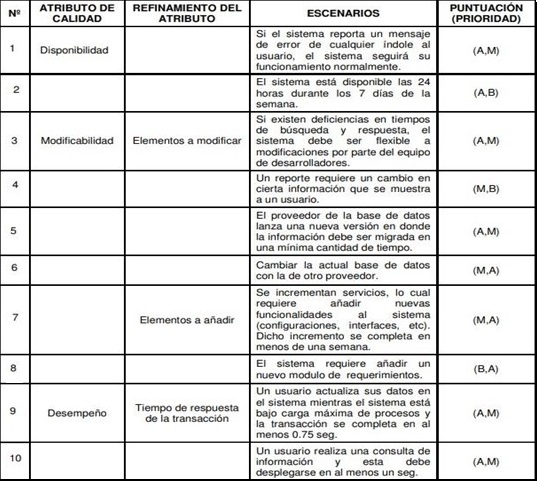
* + **Arquitectura de 3 capas**: El sistema está estructurado en tres capas principales:
* **Capa de presentación**: El frontend desarrollado en **React**, que proporciona la interfaz interactiva para los usuarios.
* **Capa lógica o de negocio**: Gestionada por el framework **Django** (Python), encargada del procesamiento de las reglas de negocio, como la gestión de citas y usuarios.
* **Capa de datos**: Gestionada por una base de datos **MySQL**, donde se almacenan todos los datos relacionados con usuarios, barberos, barberías y citas.
  + **Procesamiento de transacciones en un ambiente cliente/servidor**: El sistema maneja las transacciones de agendamiento de citas y gestión de usuarios a través de un entorno de **cliente/servidor**, donde las solicitudes se envían desde el cliente (usuario) al servidor, que procesa la solicitud y devuelve la respuesta.
  + **Arquitectura orientada a objetos**: *BarberConnect* utiliza un enfoque orientado a objetos en su diseño, lo que permite una fácil extensión y mantenimiento del código, así como una estructura modular que separa claramente las responsabilidades entre los componentes del sistema.
  + **Enlaces HTML entre módulos**: Las diferentes secciones y módulos de la plataforma (como la gestión de perfiles, citas y servicios) están conectadas mediante **enlaces HTML** y rutas bien definidas, lo que facilita la navegación y la interacción del usuario con el sistema.
  + **Arquitectura de datos centralizada**: Todos los datos del sistema están centralizados en una base de datos **MySQL**, que se considera una solución confiable y escalable para gestionar la información del sistema de manera eficiente.
  + **Estilo arquitectónico de llamada y retorno**: *BarberConnect* utiliza un estilo de **llamada y retorno** en su arquitectura, típico de las arquitecturas basadas en capas. Esto significa que las solicitudes de los usuarios generan llamadas al servidor, que procesa la lógica de negocio y devuelve la respuesta adecuada, siguiendo un ciclo claro de procesamiento.



## 5.GENERACION DEL ARBOL DE UTILIDAD:

En esta etapa se identifican, priorizan y redefinen las metas de algunos atributos de calidad de la arquitectura de *BarberConnect*. La generación del árbol de utilidad es el resultado de la priorización de los requerimientos específicos relacionados con los atributos de calidad, observados como escenarios clave del sistema.

A continuación, se presenta el árbol de utilidad generado durante la aplicación del modelo ATAM para *BarberConnect*. En este árbol, se identifican los principales atributos de calidad necesarios para la arquitectura del software, cada uno de los cuales se refina con su correspondiente valor de prioridad (Alto, Medio, Bajo).



## 6.ANALIZAR LOS ENFOQUES ARQUITECTONICOS

En este paso, se analizan los escenarios más importantes de acuerdo con el árbol de utilidad generado para *BarberConnect*, estableciendo las decisiones arquitectónicas que se deben tomar para cada escenario, junto con sus puntos de sensibilidad, riesgos, no riesgos, y tradeoffs (cuando la decisión afecta a más de una característica de calidad). Dado que algunos de los escenarios implican cambios, se analiza el impacto de estos cambios en la arquitectura del sistema.

A continuación, se detallan los puntos de sensibilidad, riesgos, no riesgos, y tradeoffs con respecto al análisis de los escenarios más importantes identificados en el árbol de utilidad de *BarberConnect*:

**Escenario 1.**

Si el sistema reporta un mensaje de error de cualquier índole al usuario, el sistema debe seguir funcionando normalmente.

* + S**1**: La **disponibilidad** del sistema es sensible al número y frecuencia de errores generados. Si no se manejan adecuadamente, los errores pueden afectar la experiencia del usuario y la operatividad del sistema. Este es un punto de sensibilidad importante que impacta negativamente la disponibilidad.
  + **R2**: Si el sistema produce errores sin un mecanismo adecuado de captura y manejo de errores, esto representa un riesgo ya que puede dejar de estar disponible para los usuarios. En *BarberConnect*, es esencial implementar un manejo robusto de errores para asegurar que el sistema siga siendo funcional y no se vea afectado por fallos menores.
  + **No riesgo**: Un adecuado manejo de errores con recuperación automática y notificaciones internas evitaría que pequeños errores afecten la **disponibilidad** del sistema.

**Escenario 2.**

Si existen deficiencias en tiempos de búsqueda y respuesta, el sistema debe ser flexible a modificaciones por parte del equipo de desarrolladores.

* **S2**: Optimizar los tiempos de búsqueda es sensible al número de elementos involucrados y la complejidad de los procedimientos. Este es un punto de sensibilidad que afecta la **modificabilidad** del sistema. Dependiendo del número de elementos a modificar, los cambios en los procedimientos pueden mejorar o empeorar la modificabilidad.
* **R3**: El acceso masivo de usuarios en un momento determinado puede generar un riesgo en los tiempos de respuesta del sistema. Si no se mantiene una calidad óptima en los algoritmos de búsqueda y en la gestión de la conectividad, los tiempos de respuesta pueden verse afectados negativamente. En *BarberConnect*, es crucial garantizar que los procedimientos de búsqueda estén optimizados para soportar altos volúmenes de solicitudes.
* **Tradeoff**: Mejorar los algoritmos de búsqueda puede aumentar la **modificabilidad**, pero puede generar un impacto en el **desempeño** y requerir una mayor optimización de los recursos del servidor.

**Escenario 3.**

El proveedor de la base de datos lanza una nueva versión en donde la información debe ser migrada en una mínima cantidad de tiempo.

* + **S3**: Cambiar a una nueva versión del motor de base de datos es sensible a la compatibilidad entre las funciones de la versión actual y las nuevas herramientas de la versión actualizada. Este es un punto de sensibilidad que afecta la **modificabilidad** del sistema, ya que los cambios en el motor de base de datos pueden afectar las consultas existentes y requerir ajustes significativos.
  + **R4**: Migrar a una nueva versión del motor de base de datos sin una planificación adecuada representa un riesgo de pérdida de datos o interrupciones en el servicio. En *BarberConnect*, es fundamental planificar y probar la migración para minimizar el tiempo de inactividad y garantizar que el sistema siga siendo operativo.
  + **Tradeoff**: La migración a una nueva versión del motor de base de datos puede mejorar el **desempeño** y la **escalabilidad**, pero también puede impactar negativamente la **modificabilidad** y aumentar la complejidad del sistema si no se gestionan adecuadamente las diferencias entre versiones.

# FASE3:

## 7.PRIORIZACION DE ESCENARIOS

En este paso, los stakeholders y el equipo de desarrollo participaron en una lluvia de ideas para identificar los escenarios más relevantes para *BarberConnect*. Se discutieron escenarios de casos de uso, escenarios de crecimiento, y escenarios exploratorios. Luego, estos escenarios se priorizaron mediante un proceso de votación en el que a cada participante se le asignó un número de votos equivalente al 30% del número total de escenarios propuestos.

El equipo de *BarberConnect*, siendo altamente productivo, generó muchos escenarios a través de este ejercicio, algunos de los cuales ya estaban incluidos en el árbol de utilidad pero no se habían analizado a fondo. Los involucrados expresaron sus opiniones sobre la importancia de los escenarios, lo que permitió que algunos de ellos adquirieran mayor relevancia que en las fases anteriores.

A continuación, se presenta una tabla con una selección de los escenarios más importantes surgidos durante este proceso de priorización. Cabe destacar que algunos escenarios no están completamente estructurados, lo que refleja la espontaneidad del ejercicio de la lluvia de ideas. Si algún escenario necesita una aclaración adicional, se realiza con la ayuda de la persona que lo propuso antes de proceder a la votación.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

.

## 8.ANALISS DE ENFOQUES:

Tras obtener los resultados de la votación de la lluvia de ideas en el paso 7, se ha encontrado que algunos de los escenarios con mayor prioridad coinciden con los descritos previamente en el árbol de utilidad. Sin embargo, también se identificaron nuevos escenarios que deben ser evaluados y considerados en la arquitectura de *BarberConnect*.

De acuerdo con los resultados del proceso de investigación y evaluación del software, se establece que la priorización del diseño debe centrarse en dos elementos clave:

* **Soporte para múltiples usuarios concurrentes**:  
  La plataforma de *BarberConnect* debe estar diseñada para soportar una gran cantidad de usuarios conectados simultáneamente. A medida que la aplicación crece y se populariza, será crucial que el sistema mantenga su capacidad de gestionar miles de reservas y consultas en tiempo real sin que esto degrade el rendimiento.
* **Conexión estable y rápida**:  
  Mantener una conexión estable y rápida es fundamental para asegurar una experiencia de usuario positiva. Los tiempos de respuesta, tanto para la búsqueda de barberías cercanas como para la confirmación de citas, deben ser mínimos. Si el sistema no puede ofrecer una experiencia fluida, los usuarios pueden optar por no utilizarlo.

# FASE4:

## 9.PRESENTACION DE RESULTADOS:

Luego de aplicar el modelo ATAM al software de *BarberConnect*, se obtuvieron las siguientes observaciones clave sobre la arquitectura y el proceso de desarrollo:

* + **Estado de la documentación del sistema**:
    - La documentación del sistema está **completa y bien organizada**. Durante la Fase 1 (Presentación) y el análisis de escenarios, se identificó que la documentación incluye detalles cruciales como:
    - Diagramas de casos de uso (vía UML).
    - Descripción detallada de las capas del sistema (capa de presentación, capa lógica y capa de datos).
    - Estructura de la base de datos, donde se especifican los modelos de usuario, barbero, cita y barbería.
* Contar con esta documentación es fundamental para asegurar que el sistema cumpla con los **requerimientos funcionales y de calidad**, así como para llevar a cabo pruebas adecuadas y garantizar un mantenimiento eficiente.
* **Evaluación arquitectónica productiva**:
  + - Gracias a la existencia de una documentación rigurosa, fue posible llevar a cabo una evaluación arquitectónica de *BarberConnect* de manera **más completa y productiva**.
    - Las entrevistas con los líderes del proyecto y el equipo de desarrollo aumentaron la cantidad de documentación generada, lo que resultó en una **mejora de la calidad arquitectónica**. La interacción constante entre los stakeholders permitió identificar áreas clave que requieren atención.
  + **Mejora de la comunicación y comprensión de requerimientos**:
    - Uno de los beneficios más destacados de la evaluación arquitectónica fue la **mejora en la comunicación** entre los diferentes involucrados (stakeholders, desarrolladores, líderes de proyecto, etc.). Esto resultó en una mejor comprensión de los requerimientos funcionales y no funcionales en los que se basa *BarberConnect*.
    - A menudo, **nuevos requerimientos** surgieron como resultado de la evaluación, lo que permitió ajustes y mejoras en la arquitectura del sistema.
  + **Uso de patrones de diseño**:
    - Durante el desarrollo del sistema, cada integrante del equipo de desarrollo se encargó de la implementación de módulos de manera independiente, utilizando **patrones de diseño** en función de sus conocimientos y la naturaleza del módulo.
    - Estos patrones de diseño ayudaron a garantizar que la arquitectura del sistema fuese modular y flexible, facilitando futuras modificaciones y el mantenimiento del sistema.
  + **Definición del diseño a alto nivel**:
    - En la arquitectura de *BarberConnect*, se enfatizó la necesidad de definir el diseño a un **nivel alto**, donde se establezcan las **restricciones**, la estructura organizacional del equipo, el razonamiento sobre posibles cambios, y una estimación clara de los **costos** y el **esfuerzo** necesario para el proyecto.
    - Este enfoque ayuda a garantizar que las decisiones arquitectónicas sean escalables y puedan manejar el crecimiento del sistema a medida que se añadan nuevas barberías o funcionalidades.
  + **Importancia del proceso de backups**:
    - Durante la evaluación, se identificó la importancia de manejar un **proceso de backups** adecuado para la base de datos central de *BarberConnect*. Esto incluye definir horarios y medios apropiados para realizar copias de seguridad.
    - Un proceso eficiente de backups garantizará la **seguridad de los datos** y la **continuidad del servicio**, ayudando a evitar la pérdida de información valiosa y contribuyendo al desempeño estable del sistema.

# 14. CONCLUCIONES*:*

Durante el desarrollo de *BarberConnect*, se ha puesto un fuerte énfasis en la investigación de tecnologías y herramientas modernas, ya que estas son relativamente nuevas para el equipo, y ninguno de los integrantes maneja de manera experta todas las tecnologías involucradas. Este proceso ha sido un desafío enriquecedor que ha permitido adquirir nuevos conocimientos y habilidades.

En un mundo que está en constante movimiento, es esencial que los proyectos evolucionen de la misma manera. En Chile, cada vez es más común ver que grandes empresas recurren a soluciones tecnológicas avanzadas, como aplicaciones móviles y plataformas web, para mantenerse conectadas con sus clientes y con las demandas del mercado. *BarberConnect* sigue esta tendencia al ofrecer una solución accesible, escalable y adaptable para gestionar citas en barberías, algo que se alinea con la evolución del entorno tecnológico actual.

Es importante destacar que el éxito del desarrollo de *BarberConnect* no se basa únicamente en el análisis y diseño arquitectónico del sistema, sino también en la creación de un **prototipo funcional**. Este prototipo ha permitido al equipo y a los interesados (profesores, stakeholders y usuarios potenciales) visualizar el progreso del sistema de una manera clara y tangible, ayudando a refinar la idea y asegurar que cumpla con los requisitos establecidos.

Teniendo en cuenta estos tres elementos clave —un buen análisis, un diseño arquitectónico sólido, y un prototipo funcional— se concluye que el enfoque metodológico adoptado ha sido adecuado para guiar el desarrollo de *BarberConnect*. Estos pilares son esenciales para garantizar que la plataforma sea capaz de escalar y evolucionar conforme aumenten los usuarios y barberías que utilicen el sistema.

En cuanto al software desarrollado, se puede concluir que las herramientas seleccionadas, como **Django** para el backend, **React** para el frontend, y **MySQL** como base de datos, han sido apropiadas. Estas tecnologías, basadas en **Python** y **JavaScript**, han permitido una fácil integración entre los diferentes módulos del sistema, proporcionando flexibilidad, escalabilidad y un desarrollo eficiente. Además, el uso de patrones de diseño orientados a objetos ha facilitado el modularidad y el mantenimiento del software.

El proceso ha sido muy enriquecedor, tanto a nivel personal como educativo, ya que nos ha permitido aplicar los conocimientos teóricos adquiridos en un proyecto práctico de mayor complejidad. Este desafío no solo ha reforzado nuestras habilidades técnicas, sino que también nos ha impuesto nuevos retos de aprendizaje, ayudándonos a ampliar nuestro dominio sobre tecnologías específicas y a enfocarnos en una sola rama de desarrollo con mayor profundidad.

Finalmente, trabajar en *BarberConnect* ha mejorado significativamente nuestras **habilidades de trabajo en equipo**. Al colaborar con distintos compañeros y líderes, hemos aprendido a valorar nuevos puntos de vista y a abordar problemas desde diferentes perspectivas. Esta dinámica ha fomentado un ambiente colaborativo y ha sido clave para el éxito del proyecto.