



**UNIVERSIDAD NACIONAL
DE COLOMBIA**



Taller 1 y 2 de proyecto

Presentado por:

Cristian Felipe Moreno Gomez

✉ crmorenogo@unal.edu.co

Diego Alejandro Rojas Reina

✉ drojasre@unal.edu.co

Edinson Sanchez Fuentes

✉ edsanchezf@unal.edu.co

Maicol Sebastian Olarte Ramirez

✉ molartera@unal.edu.co

**Facultad de Ingeniería
Departamento de Sistemas e Industrial
Ingeniería de Software 1 (2016701) - Grupo 2
2024 - 2**

Punto 1: Pre Requerimientos

SIN ESTE PUNTO NO SE REVISARÁN LOS SIGUIENTES.

En el archivo [README.md](#) del repositorio, incluyan lo siguiente:

1. **Título del repositorio:**
 - Ejemplo: Repositorio grupal - Ingeniería de Software 1 - 2024-2 Grupo #.
2. **Nombres de los integrantes del equipo.**
3. **Nombre del proyecto.**
4. **Icono o logo del proyecto (opcional).**
5. **Descripción del objetivo del proyecto:**

Descripción de lo que trata el proyecto.
6. **Tecnologías a utilizadas:**
 - Lenguajes de programación.
 - i. Javascript
 - ii. MySQL
 - Frameworks.
 - i. Next
 - ii. Express
 - Servicios externos (bases de datos, servidores, APIs).
 - i. Vercel
7. **Archivo .gitignore:**

implementar el [.gitignore](#), les recomiendo esta pagina [?](#)

8. <https://www.toptal.com/developers/gitignore>

Punto 2: Levantamiento de requerimientos

Este punto se enfoca en identificar y documentar las necesidades del proyecto desde sus fundamentos. Para estructurar este apartado, utilicen las siguientes preguntas orientadoras a para redactar su texto a [manera de narración](#):

- **¿De dónde surge la idea?**
Explicar cómo identificaron la necesidad del sistema. ¿Fue por observación, entrevistas, investigación o una combinación de estas? ¿Qué problemática busca resolver el software?
- **¿Cómo se pusieron de acuerdo para elegir la idea?**
Describan el proceso de consenso dentro del equipo: reuniones, votaciones, asignación de roles, etc.
- **¿Cuáles son las problemáticas que buscan resolver?**
Contextualicen los problemas que enfrenta el público objetivo. Justifiquen la relevancia de abordarlos con un sistema de software.
- **¿Qué expectativas tienen los usuarios potenciales del sistema?**
Identifiquen las principales funcionalidades esperadas y los beneficios que el sistema debe proporcionar.
- **¿Qué beneficios esperan ustedes al desarrollar este proyecto?**
Reflexionen sobre lo que cada integrante espera aprender y cómo el proyecto contribuye a su formación profesional.

El levantamiento debe hablar muy brevemente sobre las funcionalidades, identificarlas es el primer paso y estas deben estar conectadas con las problemáticas mencionadas.

Pcassembler surge como una necesidad a la hora de realizar...

LLuvia de ideas de requerimientos:

Pestaña perfil de usuario

Pestaña de bienvenida (presentación)

Pestaña nosotros

Pestaña PCs (Top ensamblados recomendamos)

Pestaña ensamblador (Componentes (posibilidad de integrar IA para la obtención de más información))

Carrito de compras (Ir a pagar)

Login, Sign up, Logout

Contexto

Aún en la actualidad, muchas personas consideran que ensamblar una PC sencilla es una tarea complicada y fuera de su alcance. Sin embargo, esta actividad puede ser una puerta de entrada al fascinante mundo de los computadores de escritorio, ofreciendo la oportunidad de aprender conceptos interesantes sin necesidad de tener conocimientos avanzados en computación.

¿De dónde surge?

Frecuentemente, quienes dudan en intentar expresan preocupaciones como el temor a dañar un componente o adquirir partes incompatibles entre sí. Por ello, se plantea una solución práctica y educativa: un ensamblador de PC sencillo, diseñado tanto para facilitar el proceso como para fomentar el aprendizaje, eliminando barreras y haciendo que esta experiencia sea accesible para todos.

¿Cómo se pusieron de acuerdo?

Tras analizar la idea en equipo, se concluyó que es un proyecto viable en un plazo corto y puede desarrollarse con los recursos disponibles actualmente. Además, no presenta implicaciones legales significativas y resulta atractivo para los integrantes del grupo. Si bien no aborda un tema completamente innovador, ofrece una perspectiva novedosa dentro del contexto de la asignatura, lo que lo convierte en una propuesta interesante y relevante.

¿Qué expectativas tienen los usuarios potenciales del sistema?

Desde otro punto de vista, los usuarios del sistema esperan que la aplicación cumpla su propósito de manera clara y sencilla, ofreciendo una experiencia intuitiva que permita a cualquier persona, independientemente de su nivel de conocimientos técnicos, utilizarla sin complicaciones. Es fundamental que la plataforma sea accesible desde cualquier lugar, adaptándose a diferentes dispositivos y entornos, para garantizar que los usuarios puedan interactuar con ella de forma eficiente en cualquier momento.

Descripción del Proyecto

La aplicación web tiene como objetivo permitir a los usuarios ensamblar computadoras de escritorio de manera sencilla y eficiente. El sitio será accesible para todos los usuarios, pero requerirá autenticación para interactuar con funciones avanzadas como el ensamblaje y la gestión personalizada de configuraciones. También se priorizará el diseño atractivo, el rendimiento óptimo y la facilidad de uso.

Requerimientos Funcionales

1. Autenticación y Registro

Permitir que los usuarios se registren y autenticquen en la plataforma para acceder a funciones avanzadas.

- Los usuarios registrados podrán ensamblar PCs, guardar sus configuraciones y gestionarlas.
- Los usuarios no autenticados podrán navegar libremente, pero no interactuar con las funcionalidades de ensamblaje.

2. Menú de Navegación

Diseñar un sistema de navegación que facilite el acceso a las diferentes secciones del sitio.

- Implementar un **navbar** con las siguientes pestañas:
 - **Bienvenida:** Información general sobre la app.
 - **Ensambla tu PC:** Página para seleccionar componentes y crear configuraciones personalizadas.
 - **Sobre nosotros:** Información sobre el proyecto y su equipo.
 - **Ensamblados Top:** Lista de configuraciones destacadas o populares.

3. Funcionalidad de Ensamblaje

Proporcionar herramientas para que los usuarios ensamblen PCs seleccionando componentes compatibles.

- Selección de componentes como CPU, GPU, RAM, tarjeta madre, almacenamiento, fuente de poder, entre otros.
- **Validación de compatibilidad** entre componentes (ej., sockets, potencia, tamaños). Este requisito es funcional ya que es indispensable para el propósito de la aplicación.

4. Costo Total del Ensamble

Calcular y mostrar el costo total del ensamblaje en tiempo real basándose en precios de los principales vendedores mundiales como lo es Amazon.

- Cada vez que el usuario seleccione un componente, el costo total se actualizará automáticamente.

5. Filtro y Búsqueda

Implementar un sistema de búsqueda para encontrar componentes específicos rápidamente y sin esfuerzo.

- Permitir filtrar por categoría, precio, marca y especificaciones técnicas.

6. Gestión de Ensamblados

Proporcionar un sistema para que los usuarios autenticados puedan guardar y gestionar sus configuraciones de PC.

- Visualización de ensamblados guardados en el perfil del usuario.
- Funciones para editar, eliminar o duplicar configuraciones existentes.

7. Guía de Ensamblaje

Incluir una guía práctica para que los usuarios entiendan cómo ensamblar los componentes seleccionados físicamente.

- Instrucciones claras y detalladas presentadas en pasos simples y gráficos.

8. Sobre nosotros

Proporcionar información detallada sobre el proyecto y cómo se abordó a partir de que y sus implicaciones.

- ¿Cómo surgió?
- De dónde surgió
- Información detallada

9. Ensamblajes recomendados

Mostrar unos ensamblajes recomendados por los creadores del sitio web , sea por su precio, su calidad o especificaciones.

- Proporcionar las especificaciones del equipo
- Brindar el precio aproximado
- Informar al usuario la calificación

10. Notificaciones por Correo Electrónico

Informar a los usuarios sobre sus ensamblajes mediante correo electrónico para informar que su ensamblaje ha sido completado con éxito.

- Enviar detalles del ensamblaje al correo del usuario tras completar la configuración.
-

Requerimientos No Funcionales

1. Diseño Responsivo

Garantizar que la página se visualice correctamente en distintos dispositivos y tamaños de pantalla.

2. Interfaz de Usuario Atractiva

Diseñar una interfaz moderna, agradable a la vista y fácil de navegar para mejorar la experiencia del usuario.

3. Rendimiento y Escalabilidad

Optimizar el sitio para soportar un alto volumen de usuarios sin afectar su velocidad ni funcionalidad.

4. Accesibilidad

Asegurar que la aplicación sea utilizable por personas con discapacidades siguiendo pautas de accesibilidad (WCAG).

Punto 3: Análisis de requerimientos

Realicen un análisis detallado de cada funcionalidad utilizando el método **MoSCoW (Must, Should, Could, Won't)**. Además, para cada funcionalidad, añadan una **estimación de esfuerzo con la secuencia de Fibonacci** (1, 2, 3, 5, 8, 13, etc.). Argumenten por qué llegaron a cada estimación considerando factores como:

- Complejidad técnica.
- Recursos disponibles.
- Impacto en la experiencia del usuario.
- Relación con los objetivos del proyecto.

NOTA: utilice tablas, gráficos, alguna técnica para organizar la información, piensen que le van a presentar esto a alguien que los quiere contratar...

Análisis de requerimientos

Requerimientos funcionales

Gestión de autenticación y registro

Gestión de autenticación y registro			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Registro de usuarios	Must Have	Implementar el registro con validaciones, manejo de errores y almacenamiento en MySQL requiere configuración del backend y frontend.	
Inicio de sesión de correo y contraseña	Must Have	Es esencial para la autenticación de usuarios y su integración con el sistema de ensamblaje y perfil.	
Recuperación de contraseña	Should Have	Aunque no es imprescindible, mejora la experiencia del usuario. Implica manejo de tokens y envíos de correo.	
Sistemas de roles(admin/usuario)	Should Have	Permitirá a los administradores	

		gestionar contenido, pero implica lógica adicional en permisos y autenticación	
--	--	--	--

Gestión de navegación

Gestión de navegación			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Implementar un navbar accesible (Inicio, Ensamble, Sobre nosotros)	Must Have	Fundamental para la navegación y accesibilidad del sistema.	

Gestión de funcionalidad de ensamblaje

Gestión de funcionalidad de ensamblaje			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Selección de componentes	Must Have	Es la base del sistema y requiere lógica para manejar compatibilidad de componentes.	
Validación de compatibilidad	Must Have	Implementar lógica de validación (ej. socket de CPU y motherboard) es complejo y requiere estructuras de datos bien definidas.	
Visualización en tiempo real de ensamblaje	Should Have	Requiere manejo dinámico del DOM y actualización en tiempo real, aumentando la complejidad.	
Recomendación de componentes según compatibilidad	Could Have	Mejoraría la UX, pero requiere una base de datos bien estructurada y lógica	

		avanzada.	
--	--	-----------	--

Gestión de costo

Gestión de costo			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Cálculo dinámico del precio	Must Have	La actualización del costo total en tiempo real es crucial para la UX y es relativamente fácil de hacer.	
Integración con API de Amazon	Must Have	Brindaría datos en tiempo real, pero depender de APIs externas y manejar su respuesta añade complejidad.	
Comparación de precios entre diferentes vendedores	Could Have	Aporta valor al usuario, pero implica manejar múltiples fuentes de datos.	

Gestión de filtro y búsqueda

Gestión de filtro y búsqueda			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Búsqueda de componentes	Must Have	Implementar búsqueda en base de datos con filtros básicos es manejable con MySQL y Next.js.	
Filtros avanzados por categoría, precio y marca	Should Have	Mejora la experiencia, pero requiere un sistema bien optimizado de queries y UI reactiva.	

Gestión de ensamble

Gestión de ensamble			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Guardar configuración de ensamblaje	Should Have	Esencial para los usuarios autenticados, pero requiere base de datos bien estructurada.	
Editar/Eliminar ensamblajes guardados	Should Have	Aumenta la funcionalidad, pero implica validaciones y seguridad en la manipulación de datos.	
Compartir ensamblajes con otros usuarios	Could Have	Facilita la colaboración, pero requiere generar links o IDs únicos.	

Gestión de guía de ensamblaje

Guía de ensamblaje			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Guía paso a paso con imágenes	Should Have	Aporta valor, pero requiere diseño gráfico y estructuración clara.	
Animaciones interactivas	Won't Have	Sería interesante, pero el tiempo y recursos limitados hacen que no sea viable en un mes.	

Gestión de ensambles recomendados

gestión de ensambles recomendados			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Personalización de recomendaciones	Could Have	Requiere un sistema de sugerencias basado en preferencias del usuario, aumentando la complejidad.	

Gestión de notificaciones

Gestión de notificaciones			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Envío de confirmacion de ensamblaje	Could Have	Agrega valor, pero depende de una API de correo como SendGrid o Nodemailer.	
Notificaciones de cambios en precios de componentes	Could Have	Útil, pero implica monitoreo de precios en la base de datos o API de terceros.	

Requerimientos no funcionales

Gestión de navegación

Diseño Responsivo			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Diseño responsivo en	Must Have	Asegura accesibilidad	

la navegación		en distintos dispositivos, pero requiere pruebas y ajustes CSS.	
---------------	--	---	--

Gestión de rendimiento y escalabilidad

Gestión de rendimiento y escalabilidad			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Agregar animaciones y transiciones suaves para mejorar la UX.	Could Have	Mejora la experiencia del usuario sin afectar la funcionalidad base.	
Optimizar carga de imágenes y componentes con lazy loading.	Could Have	Reduce tiempos de carga, pero no es crítico para el MVP.	

Gestión de accesibilidad

Gestión de accesibilidad			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Cumplir parcialmente con las pautas WCAG (Web Content Accessibility Guidelines) .	Could Have	Permite que personas con discapacidades usen la app, pero no es esencial para la funcionalidad base.	

Gestión de seguridad

Gestión de seguridad			
Funcionalidad	Prioridad (MoSCoW)	Descripción	Estimación (Esfuerzo)
Protección contra ataques de inyección SQL en consultas de MySQL.	Should Have	Evita vulnerabilidades de seguridad en la base de datos.	

Punto 4: Análisis gestión de software

Analicen la **triada de gestión de proyectos** para su proyecto. Investigar y documentar lo siguiente:

- **Tiempo:**
¿Cuánto tiempo estiman que tomará el desarrollo de cada funcionalidad o módulo? Detallan cómo se dividirán las etapas del desarrollo (por ejemplo, diseño, desarrollo, pruebas).
- **Costo:**
Investigar cuánto cuesta desarrollar software considerando factores como:
 - Sueldos de desarrolladores (junior, senior, arquitectos, ux, testers).
 - Licencias, servicios en la nube, bases de datos.
 - Herramientas externas (APIs, frameworks, software de desarrollo).
 - Infraestructura (servidores, almacenamiento).

Saquen un total en una tabla organizada partiendo en que solo mostrarán lo que necesiten dentro de los costos (por ejemplo solo ocupamos un desarrollador junior)

- **Alcance:**
Definan los límites del sistema. ¿Qué características o funcionalidades están incluidas y cuáles quedan fuera del alcance del MVP? Sean claros para evitar ambigüedades en futuras iteraciones del proyecto.

NOTA: Aunque esto no lo vimos en clase, lo que quiero que intenten es que investiguen en internet cuánto vale realmente su trabajo, esto lo deben siempre en el mundo laboral real, es la intención del ejercicio, no usen IA, o terminarán regalándose por 50k a la semana (?).

Para abordar este análisis de gestión de software se va a analizar cada módulo en general tomando como base lo realizado en el punto anterior es decir cada funcionalidad que de este se desglosa, se estima el módulo por su tiempo en días de diseño, desarrollo y pruebas, para luego hallar un total de días, y con esto generar los costos en cuanto a nómina.

Se debe tener en cuenta que para este análisis se considera un equipo de cuatro personas, tres programadores junior con muy poco tiempo de experiencia y un programador semi-senior el cual está encargado de dirigir el proyecto.

Módulo	Diseño	Desarrollo	Pruebas	Total
Gestión de autenticación y registro	1	2	1	4

Gestión de navegación	0.5	1	0.5	2
Gestión de funcionalidad de ensamblaje	1.5	6.5	1	9
Gestión de costo	0.25	0.5	0.25	1
Gestión de filtro y búsqueda	0.25	0.5	0.25	1
Gestión de ensamblaje	1	3	1	5
Guía de ensamblaje	1.75	2	0.25	4
Gestión de ensamblajes recomendados	1	2	1	4
Gestión de notificaciones	0.25	0.5	0.25	1
Diseño Responsivo	2	3	1	6
Gestión de rendimiento y escalabilidad	1	2	1	4
Gestión de accesibilidad	1	1	1	3
Gestión de seguridad	1	4	1	5
Total	12.5	28	9.5	49

Ahora a partir de esto se relaciona el equipo de trabajo disponible para ejecutar los anteriores módulos, el salario base mensual se fija de acuerdo a glassdoor y a indeed.

Nombre	Rol	Salario Base Mensual
Diego Alejandro Rojas Reina ✉ drojasre@unal.edu.co	Programador Semi-Senior	\$4,500,000
Cristian Felipe Moreno Gomez ✉ crmorenogo@unal.edu.co	Programador Junior	\$2,700,000
Edinson Sanchez Fuentes ✉ edsanchezf@unal.edu.co	Programador Junior	\$2,700,000
Sebastian Olarte Ramirez ✉ molartera@unal.edu.co	Programador Junior	\$2,700,000

Por otro lado en cuanto a recursos como almacenamiento en la nube, APIs , servidores y demás se contempla lo siguiente, cabe aclarar que para el ejercicio se contemplan precios asimilando que el

software está en operación , recibiendo una cantidad considerable de tráfico y con todos los servicios en la nube, pero para el caso del curso se optara por todos los planes gratuitos.

Digitalocean para alojar la base de datos.

The screenshot shows the DigitalOcean website's pricing section. On the left, there's a navigation menu with links like 'Volver a la descripción general', 'Niveles', 'Precios de contenedores de plataformas de aplicaciones', 'Precios adicionales de la plataforma de aplicaciones', and 'Preguntas frecuentes'. The main content area displays two pricing plans side-by-side.

Nivel gratuito	Nivel de pago
A partir de \$0 /mes	A partir de \$5 /mes
<ul style="list-style-type: none"> ✓ 3 aplicaciones con sitios estáticos ✓ 1 GiB de transferencia de datos por aplicación con sitios estáticos ✓ Implementación desde GitHub y Gitlab ✓ HTTPS automático ✓ Trae tu dominio personalizado ✓ CDN global ✓ Mitigación de DDoS ✓ Miembros del equipo ilimitados 	<p>Incluye todas las funciones del nivel gratuito más:</p> <ul style="list-style-type: none"> ✓ Implementación desde registros de contenedores ✓ CPU compartida y dedicada ✓ Escalamiento horizontal y vertical ✓ Escalado automático basado en CPU ✓ Aplicación automática de parches al sistema operativo ✓ Métricas de aplicaciones por hora ✓ Reenvío de registros ✓ Alta disponibilidad ✓ IP de salida dedicada ✓ Bases de datos de desarrollo y producción ✓ Hasta 10 revisiones para reversiones

Auth0 para autenticación

The screenshot shows the Auth0 website's pricing section. It features two main plans: a free plan and a paid plan labeled 'B2C - ESSENTIALS'.

Plan gratuito	B2C - ESSENTIALS
Obtenga Auth0 gratis con hasta 7500 usuarios activos e inicios de sesión ilimitados. No se necesita tarjeta de crédito.	Para proyectos básicos o pequeñas aplicaciones.
Empezar a construir con Auth0 →	\$35 /mo
	1000 tokens M2M
	1 propietario de producción
	Hasta 7 administradores de cuentas
	Conexiones sociales ilimitadas
	SDK, proveedores de identidad de Auth0 y pantalla de bloqueo
	Hasta 5 acciones

Google cloud tiene un inicio gratuito pero también en su plan más económico cuesta lo siguiente.

	<h1>STARTER</h1> <p>Ideal para pequeñas empresas o proyectos individuales que requieren atención básica</p>
PRECIO MENSUAL	USD \$50.00 + 3% consumo

Recursos	Estimado Mensual	Descripción
Base de Datos (MySQL)	\$20,681.82	AWS RDS, DigitalOcean, o Google Cloud SQL.(Hay planes gratuitos)
API de Autenticación	\$ 144,772.75	Auth0 o Firebase (plan gratuito hasta cierto número de usuarios)
Email.JS	\$ 0	Ofrece un plan gratuito hasta cierto número de correos al día.
Hosting (Backend)	\$20,681.82	AWS, DigitalOcean, o Heroku
Almacenamiento en la Nube	\$206,818.22	S3, Google Cloud Storage, DigitalOcean Spaces
Servicios de APIs Externas	\$ 0	APIs de terceros, como Amazon Product Advertising(Gratuita hasta cierto punto)
Total	\$392,954.61	

Ahora para calcular el total se relaciona el número de días que va tomar el proyecto y los servicios mencionados para despliegue, es decir para el primer mes.

Item	Valor	Cantidad	Total
Programador Semi-Senior	\$150,000.000	1	\$7,350.000.000
Equipo Programadores Junior	\$90,000.000	1	\$13,230.000.000
Recursos	\$392,954.61	1	\$392,954.61
Total			\$20,972.954.61

Punto 5: Casos de uso del sistema

Elaboren casos de uso exclusivamente para las funcionalidades clasificadas como **MUST** en el análisis MoSCoW. Cada estudiante deberá realizar un mínimo de **tres casos de uso** es decir, si no son suficientes los MUST, utilicen los SHOULD.

Documenten los casos de uso con los siguientes elementos como mínimo:

- Nombre del caso de uso.
- Descripción breve.
- Actor(es) involucrado(s).
- Flujo principal.
- Flujos alternativos (si aplica).
- Precondiciones y postcondiciones (si aplica).

Además, complementan los casos de uso con su diagrama (**ES SÓLO UNO POR EL SISTEMA**), para ambos grupos su diagrama tiene que verse parecido a esto... [URL](#) aquellos que se equivocaron en el taller, espero lo corrijan, los revisaré con detalle ☹☹

NOTA: Sé que anteriormente había dicho que todas las funcionalidades (MoSCoW) pero hubo grupos que han hecho demasiados casos de uso, la idea es que aprendan a como hacerlos, no ahogarlos en repetir el proceso, a los que hayan avanzado y hayan hecho más de los que les pido, me excuso desde ya.

Casos

de

uso:

Registrar nuevo usuario**Actor:** Persona no registrada**Breve descripción:**

El usuario proporciona sus datos personales para crear una cuenta en la plataforma. El sistema valida la información y almacena el registro en la base de datos si todos los datos son correctos.

Descripción Paso a Paso**Precondiciones:**

- El usuario no debe estar registrado en el sistema con el mismo correo electrónico.
- Debe contar con conexión a Internet para completar el proceso.

Flujo principal:

1. El usuario accede a la opción de "Registrarse".
2. El sistema muestra un formulario con los siguientes campos:
 - Nombre de usuario.
 - Correo electrónico.
 - Contraseña.
3. El usuario completa los campos requeridos y envía el formulario.
4. El sistema realiza la validación de los datos ingresados:
 - **Nombre de usuario:**
 - Debe contener al menos 4 caracteres.
 - Si no cumple con este requisito, se solicita la corrección.
 - **Contraseña:**
 - Mínimo 8 caracteres.
 - Debe contener al menos una letra mayúscula, una minúscula y un carácter especial.
 - Si no cumple con estos requisitos, se informa al usuario sobre las correcciones necesarias.
 - **Correo electrónico:**
 - Debe seguir el formato válido (ejemplo@dominio.com).
 - Si el formato es incorrecto, se solicita la corrección.
5. El sistema verifica si el correo ya está registrado:
 - Si ya existe una cuenta con ese correo, se muestra un mensaje de error.
 - Si no está registrado, el proceso continúa.
6. Si todas las validaciones son exitosas, el sistema guarda los datos del usuario en la base de datos.
7. Se muestra un mensaje de confirmación y el usuario puede iniciar sesión.

Flujos alternativos:

- **(4a) Datos inválidos:** Si alguno de los datos ingresados no cumple con los requisitos, el sistema muestra mensajes de error específicos y solicita la corrección antes de continuar.
- **(5a) Correo ya registrado:** Si el correo ingresado ya existe en la base de datos, se informa al usuario y se le da la opción de recuperar su cuenta.

- **(6a) Error en el servidor:** Si ocurre un problema al registrar la cuenta, se muestra un mensaje de error y se solicita intentar nuevamente más tarde.

Postcondiciones:

- Si el proceso fue exitoso, el usuario queda registrado y puede autenticarse en el sistema.
- Si el proceso falla, los datos no se almacenan y el usuario debe intentarlo nuevamente.

Iniciar sesión
Actor: Usuario registrado
Breve descripción: El usuario ingresa sus credenciales para acceder a su cuenta en la plataforma.
Descripción Paso a Paso <p>Precondiciones:</p> <ul style="list-style-type: none"> • El usuario debe estar registrado en el sistema. • Debe contar con conexión a Internet para autenticarse. <p>Flujo principal:</p> <ol style="list-style-type: none"> 1. El usuario accede a la opción "Iniciar sesión". 2. El sistema muestra un formulario con los siguientes campos: <ul style="list-style-type: none"> ○ Correo electrónico. ○ Contraseña. 3. El usuario ingresa sus credenciales y envía el formulario. 4. El sistema valida los datos ingresados: <ul style="list-style-type: none"> ○ Correo electrónico: <ul style="list-style-type: none"> ■ Debe tener el formato correcto. ■ Si es incorrecto, se solicita la corrección. ○ Contraseña: <ul style="list-style-type: none"> ■ Debe coincidir con la registrada en la base de datos. ■ Si es incorrecta, se muestra un mensaje de error. 5. Si las credenciales son correctas, el sistema permite el acceso a la cuenta. 6. El usuario es redirigido a la página principal de su perfil. <p>Flujos alternativos:</p> <ul style="list-style-type: none"> • (4a) Credenciales incorrectas: Si el correo o la contraseña no son válidos, se muestra un mensaje de error y se solicita intentar nuevamente. • (4b) Cuenta no registrada: Si el correo no está asociado a una cuenta, se informa al usuario y se ofrece la opción de registrarse. • (5a) Problema en el servidor: Si ocurre un error en la autenticación, el sistema muestra un mensaje indicando que intente más tarde. <p>Postcondiciones:</p> <ul style="list-style-type: none"> • Si el inicio de sesión es exitoso, el usuario accede a su cuenta. • Si el inicio de sesión falla, el usuario debe corregir los datos o registrarse.

Implementar Navbar accesible**Actor:** Usuario (registrado o no registrado)**Breve descripción:**

El usuario navega entre las diferentes secciones de la plataforma a través del menú de navegación.

Descripción Paso a Paso**Precondiciones:**

- El usuario debe tener acceso a la plataforma.
- La interfaz debe estar cargada correctamente.

Flujo principal:

1. El usuario accede a la plataforma.
2. El sistema muestra el Navbar con las siguientes opciones:
 - Inicio.
 - Ensamble.
 - Sobre nosotros.
3. El usuario selecciona una opción del menú.
4. El sistema redirige al usuario a la página correspondiente.

Flujos alternativos:

- **(2a) Error en la carga del Navbar:** Si el menú no carga correctamente, se muestra un mensaje de error y se intenta recargar la página.
- **(3a) Enlace roto:** Si la redirección falla, el usuario recibe un mensaje de error y puede intentar nuevamente.

Postcondiciones:

- El usuario accede a la sección deseada sin problemas.
- En caso de error, puede intentarlo nuevamente.

Selección de Componentes**Actor:** Usuario autenticado**Breve descripción:**

El usuario escoge los componentes de su PC, seleccionando opciones compatibles con el ensamble.

Descripción Paso a Paso**Precondiciones:**

- El usuario debe haber iniciado sesión.
- Debe existir una lista de componentes disponibles.

Flujo principal:

1. El usuario accede a la sección "Ensambla tu PC".
2. Se muestra una lista de categorías de componentes (CPU, GPU, RAM, etc.).
3. El usuario selecciona un componente para cada categoría.
4. El sistema valida la compatibilidad entre los componentes seleccionados.
5. Si todo es compatible, se permite continuar con la configuración.

Flujos alternativos:

- **(4a) Componente incompatible:** Si se selecciona un componente no compatible, se muestra un mensaje de error y se solicita elegir otro.
- **(5a) Error en la carga de datos:** Si ocurre un problema al cargar los componentes, el sistema muestra un mensaje y solicita intentarlo más tarde.

Postcondiciones:

- Si el ensamble es válido, se permite avanzar.
- Si hay problemas de compatibilidad, se solicita corregirlos.

Validación de Compatibilidad
Actor: Usuario autenticado
Breve descripción: El sistema verifica que los componentes seleccionados sean compatibles antes de completar el ensamble.
Descripción Paso a Paso Precondiciones: <ul style="list-style-type: none">• El usuario debe haber seleccionado componentes para su PC. Flujo principal: <ol style="list-style-type: none">1. El usuario finaliza la selección de componentes.2. El sistema revisa la compatibilidad basándose en reglas predefinidas (ej., socket de CPU y motherboard).3. Si todo es compatible, el sistema muestra un mensaje de éxito.4. Se habilita la opción para guardar la configuración. Flujos alternativos: <ul style="list-style-type: none">• (2a) Incompatibilidad detectada: Si hay un problema, el sistema notifica al usuario e indica qué componente debe cambiar.• (3a) Error en la validación: Si el sistema no puede verificar la compatibilidad, se solicita intentarlo nuevamente. Postcondiciones: <ul style="list-style-type: none">• Si el ensamble es válido, el usuario puede guardarlo.• Si hay incompatibilidades, el usuario debe corregirlas.

Cálculo dinámico del precio**Actor:** Usuario autenticado**Breve descripción:**

El sistema calcula en tiempo real el costo total del ensamble con base en los componentes seleccionados.

Descripción Paso a Paso**Precondiciones:**

- El usuario debe haber seleccionado al menos un componente.
- Deben existir precios asociados a cada componente.

Flujo principal:

1. El usuario selecciona un componente.
2. El sistema obtiene el precio del componente y lo suma al costo total.
3. Cada vez que se agrega o elimina un componente, el sistema actualiza el total.
4. El precio total se muestra en la interfaz en tiempo real.

Flujos alternativos:

- **(2a) Precio no disponible:** Si el precio de un componente no está disponible, se muestra un mensaje y no se incluye en el cálculo.
- **(3a) Error en la actualización:** Si hay problemas al calcular el precio, el sistema muestra un mensaje de error.

Postcondiciones:

- El usuario ve el costo total actualizado en tiempo real.
- Si hay errores, el usuario recibe una notificación.

Integración con API de Amazon**Actor:** Usuario autenticado**Breve descripción:**

El sistema obtiene información en tiempo real de los precios y disponibilidad de componentes desde Amazon.

Descripción Paso a Paso**Precondiciones:**

- El sistema debe estar conectado a la API de Amazon.

Flujo principal:

1. El usuario accede a la sección de ensamblaje.
2. El sistema consulta la API de Amazon para obtener los precios y disponibilidad de los componentes.
3. Los datos obtenidos se muestran en la interfaz del usuario.

Flujos alternativos:

- **(2a) Error en la conexión con la API:** Si la API de Amazon no responde, el sistema muestra un mensaje y utiliza los últimos datos almacenados.
- **(3a) Precio no encontrado:** Si Amazon no tiene información sobre un componente, se muestra como "No disponible".

Postcondiciones:

- Si la integración es exitosa, el usuario verá información actualizada de precios.
- Si hay errores, se usan datos previos o se notifica al usuario.

Búsqueda de componentes
Actor: Usuario registrado o no registrado
Breve descripción: El usuario puede buscar componentes en la base de datos utilizando filtros básicos para facilitar la selección.
Descripción Paso a Paso Precondiciones: <ul style="list-style-type: none">• Debe existir una base de datos con componentes almacenados.• El sistema debe permitir la búsqueda eficiente. Flujo principal (escenario ideal): <ol style="list-style-type: none">1. El usuario accede a la sección de búsqueda de componentes.2. El sistema muestra una barra de búsqueda y filtros disponibles (categoría, marca, precio, disponibilidad, etc.).3. El usuario ingresa un término de búsqueda o selecciona filtros.4. El sistema consulta la base de datos y muestra los resultados coincidentes.5. El usuario selecciona un componente y accede a sus detalles. Flujos alternativos: <ul style="list-style-type: none">• (4a) No hay resultados: Si la búsqueda no arroja coincidencias, el sistema muestra un mensaje indicando que no hay componentes disponibles.• (4b) Problema en la consulta: Si ocurre un error en la búsqueda, el sistema muestra un mensaje de error e invita a intentarlo nuevamente. Postcondiciones: <ul style="list-style-type: none">• Si la búsqueda es exitosa, el usuario visualiza los resultados.• Si no hay coincidencias, el usuario puede modificar los filtros o términos de búsqueda.

Diseño responsivo en la navegación
Actor: Usuario registrado o no registrado
Breve descripción: El sistema adapta la interfaz de navegación a distintos dispositivos (computadoras, tabletas y móviles) para mejorar la accesibilidad.
Descripción Paso a Paso Precondiciones: <ul style="list-style-type: none">• El sistema debe tener un diseño adaptable implementado en CSS y/o frameworks responsivos. Flujo principal (escenario ideal): <ol style="list-style-type: none">1. El usuario accede a la plataforma desde cualquier dispositivo.2. El sistema detecta el tamaño de pantalla y ajusta la interfaz automáticamente.3. El usuario navega por el sistema sin dificultades, sin importar el tamaño de su pantalla. Flujos alternativos: <ul style="list-style-type: none">• (2a) Fallo en la adaptación: Si el sistema no se ajusta correctamente a un dispositivo, se registra el problema y se muestra una opción de "Ver en versión de escritorio". Postcondiciones: <ul style="list-style-type: none">• Si el diseño es responsivo, el usuario puede interactuar cómodamente con la plataforma.• En caso de errores, se brinda una alternativa de navegación.

Recuperación de contraseña**Actor:** Usuario registrado**Breve descripción:**

Permite a los usuarios restablecer su contraseña en caso de olvido mediante el uso de tokens y correo electrónico.

Descripción Paso a Paso**Precondiciones:**

- El usuario debe tener una cuenta registrada con un correo válido.
- El servidor debe estar configurado para enviar correos electrónicos.

Flujo principal:

1. El usuario accede a la opción "¿Olvidaste tu contraseña?".
2. Se le solicita ingresar su correo electrónico.
3. El sistema verifica que el correo esté registrado.
4. Si el correo es válido, el sistema genera un token de recuperación y lo envía por correo.
5. El usuario recibe un enlace en su correo y accede a la página de restablecimiento.
6. Ingresa una nueva contraseña cumpliendo con los requisitos de seguridad.
7. El sistema valida la nueva contraseña y la actualiza en la base de datos.
8. Se notifica al usuario que su contraseña ha sido restablecida.

Flujos alternativos:

- **(3a) Correo no registrado:** Si el correo ingresado no existe en la base de datos, se muestra un mensaje de error.
- **(5a) Token expirado:** Si el usuario intenta usar un enlace caducado, se le solicita generar uno nuevo.
- **(7a) Contraseña inválida:** Si la nueva contraseña no cumple los requisitos, se solicita corregirla.

Postcondiciones:

- Si el proceso es exitoso, el usuario podrá iniciar sesión con su nueva contraseña.
- Si hay errores, el usuario debe corregirlos antes de continuar.

Sistemas de roles (admin/usuario)
Actor: Administrador, usuario registrado
Breve descripción: El sistema diferencia los permisos entre usuarios normales y administradores, permitiendo a los administradores gestionar contenido.
Descripción Paso a Paso Precondiciones: <ul style="list-style-type: none">• Un usuario debe estar autenticado en el sistema.• Debe existir una base de datos con roles definidos. Flujo principal: <ol style="list-style-type: none">1. El usuario inicia sesión.2. El sistema identifica su rol en la base de datos.3. Si el usuario es administrador, se habilitan opciones adicionales de gestión (gestionar componentes, usuarios, pedidos, etc.).4. Si el usuario es estándar, solo puede acceder a las funciones de ensamblaje y visualización. Flujos alternativos: <ul style="list-style-type: none">• (2a) Usuario sin permisos: Si un usuario intenta acceder a funciones de administrador sin los permisos adecuados, se le deniega el acceso.• (3a) Error en la carga de roles: Si ocurre un problema en la identificación del rol, se muestra un mensaje de error. Postcondiciones: <ul style="list-style-type: none">• Los administradores pueden gestionar el sistema según sus permisos.• Los usuarios normales solo pueden acceder a funciones básicas.

Visualización en tiempo real de ensamblaje
Actor: Usuario autenticado
Breve descripción: El sistema muestra una vista dinámica del ensamblaje de la PC mientras el usuario selecciona los componentes.
Descripción Paso a Paso Precondiciones: <ul style="list-style-type: none"> • El usuario debe haber iniciado sesión. • Debe existir una interfaz gráfica interactiva para visualizar el ensamblaje. Flujo principal: <ol style="list-style-type: none"> 1. El usuario accede a la sección de ensamblaje. 2. Elige un componente (CPU, GPU, RAM, etc.). 3. El sistema actualiza la vista del ensamblaje en tiempo real. 4. Cada vez que se agrega o cambia un componente, la interfaz se ajusta dinámicamente. 5. Si el ensamblaje es compatible, se permite guardar la configuración. Flujos alternativos: <ul style="list-style-type: none"> • (3a) Error en la actualización: Si el sistema no puede actualizar la vista, se muestra un mensaje de error y se solicita intentarlo nuevamente. • (4a) Componente incompatible: Si un componente no es compatible con el ensamblaje actual, se muestra un mensaje de advertencia. Postcondiciones: <ul style="list-style-type: none"> • El usuario puede ver su ensamble en tiempo real. • Si hay errores, puede corregirlos antes de finalizar la selección.

Punto 6: Historias de usuario

Para cada caso de uso generado en el punto anterior, cada estudiante debe crear su respectiva **historia de usuario**, con un mínimo de **tres historias por persona**. Utilicen la plantilla proporcionada. Hasta ahora, todos utilizan correctamente la plantilla.

6.1 HU Login

Anexo de Documentos Relacionados:

- En este apartado se agregan toda la documentación que sea relevante para entender o sustentar la historia de usuario

Descripción conceptual

Módulo	Autenticación y autorización de usuarios
Descripción de la(s) funcionalidad(es) requerida(s):	El sistema debe identificar y diferenciar entre los roles de administrador y cliente mediante un registro/ login seguro para personalizar las funcionalidades disponibles.

Descripción técnica

En este apartado existen dos partes, la del backend y la del frontend

Backend:

URL	Método	Código html
localhost:8080/login	GET	200 401
Caso de uso técnico Al consultar, debe retornar código 200 y en data un array con los datos de los usuarios si retorna 401 es porque las credenciales están mal		
Datos de entrada { "data": [{ "usuario": "Juan Pérez" "password": "***** " }] }	Datos de salida { "status": "success", "data": [{ "id": 1, "nombre": "Juan Pérez" "imagen": "imagen.html" }] }	
Datos de entrada { "data": [{ "usuario": "Juan Pérez" "password": "***** " }] }	Datos de salida { "status": "fail", "data": [{ "message": "credenciales incorrectas", }] }	

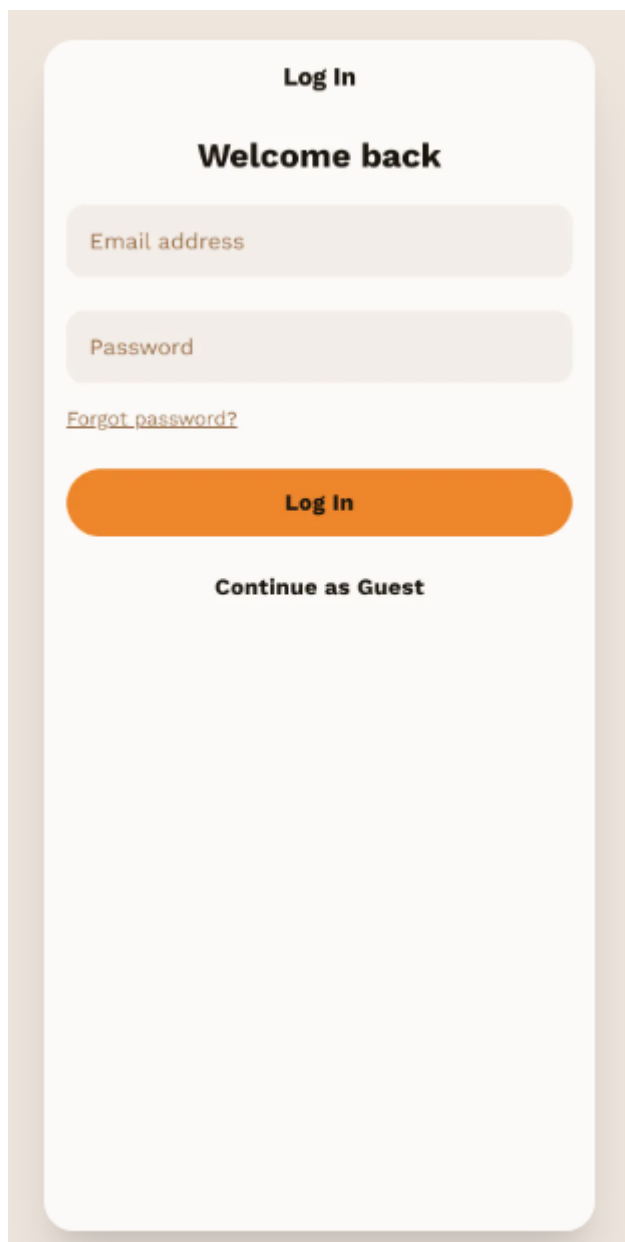
Frontend

En este apartado se debe describir no solo con mockups, si no con una descripción narrada lo que debe hacer la HU funcionalmente o visualmente, detalles para que hagan una buena descripción

Interacción esperada:

el usuario se meterá a esa vista y diligenciara sus credenciales

Mockups/Prototipos:

A mobile app mockup of a login screen. The screen has a light beige background. At the top, the text "Log In" is centered in a bold, black font. Below it, "Welcome back" is also centered in a bold, black font. There are two input fields: "Email address" and "Password", both with light beige backgrounds and rounded corners. Below the password field is a link that says "Forgot password?" in a smaller, underlined font. At the bottom, there is a large, rounded orange button with the text "Log In" in white. Below the button, the text "Continue as Guest" is centered in a bold, black font.

6.2 HU Registro

Anexo de Documentos Relacionados:

- En este apartado se agregan toda la documentación que sea relevante para entender o sustentar la historia de usuario

Descripción conceptual

Módulo	<i>Registro de usuarios</i>
Descripción de la(s) funcionalidad(es) requerida(s):	<i>El sistema debe permitir al usuario crear un perfil</i>

Descripción técnica

En este apartado existen dos partes, la del backend y la del frontend

Backend:

URL	Método	Código html
localhost:8080/register	GET	200 400
Caso de uso técnico Al consultar, debe retornar código 200 y en data un array con los datos de los usuarios si retorna 400 es no se puede crear el usuario debido a una Bad Request		
Datos de entrada <pre>{ "data": [{ "usuario": "Juan Pérez" "password": "*****" "imagen": "imagen.jpg" }] }</pre>	Datos de salida <pre>{ "status": "success", "data": [{ "id": 1, "nombre": "Juan Pérez" "imagen": "imagen.jpg" }] }</pre>	
Datos de entrada <pre>{ "data": [{ "usuario": "J" "password": "*****" }] }</pre>	Datos de salida <pre>{ "status": "fail", "data": [{ "message": "Bad Request", }] }</pre>	

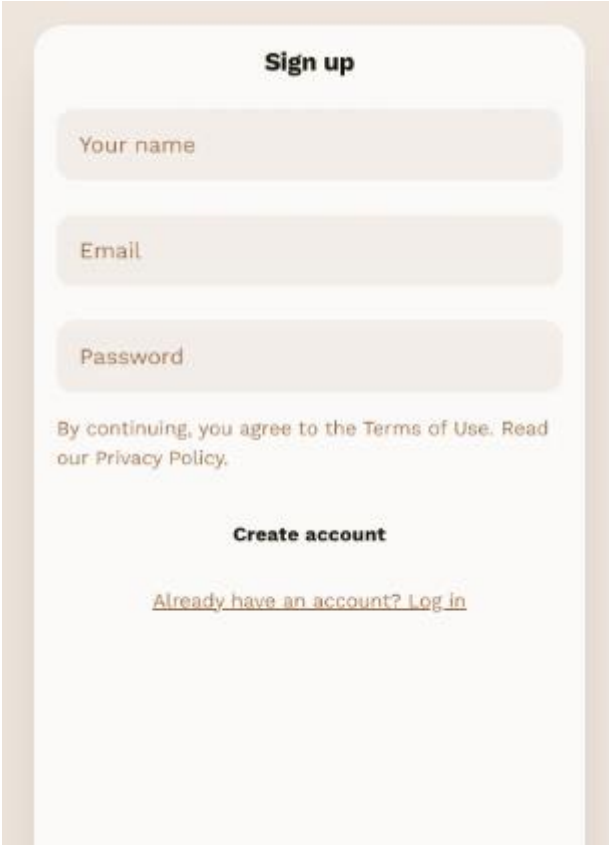
Frontend

En este apartado se debe describir no solo con mockups, si no con una descripción narrada lo que debe hacer la HU funcionalmente o visualmente, detalles para que hagan una buena descripción

Interacción esperada:

(Describir cómo el usuario interactúa con la funcionalidad, ejemplo: un botón, un formulario, un modal, etc.)

Mockups/Prototipos:

A mockup of a 'Sign up' form. The form is titled 'Sign up' in bold. It contains three input fields: 'Your name', 'Email', and 'Password'. Below the input fields, there is a line of text: 'By continuing, you agree to the Terms of Use. Read our Privacy Policy.' Below this text is a button labeled 'Create account'. At the bottom, there is a link that says 'Already have an account? Log in'.

Flujo visual y eventos:

el usuario creara su perfil llenado el email, password y el nombre del usuario

A partir de aquí inicia la parte 2 del taller.

Punto 7: Clean Code

Tal como les contaba en clase, existe automatización para generar informes de clean code, para cada uno de los frameworks utilizados (backend y frontend, kotlin para los de móvil)

- Seleccionar e implementar **linters** y/o herramientas de análisis estático de código según el lenguaje de programación utilizado (por ejemplo, ESLint, SonarQube, Pylint, etc.).
- Realizar el paso a paso de implementación con pantallazos mostrando el proceso para automatizar en el proyecto
- Ejecutar la herramienta elegida, y tomar la captura de pantalla de su resultado.
- Cada integrante debe hacer una narración corta y muy breve sobre cómo han visto el código de sus compañeros, ¿fue fácil de leer? ¿sabes en qué trabajaba tu compañero? ¿si entras a modificarlo tendrías líos? ¿cumple el estándar que designaron?

Punto 8: Diseño y Arquitectura

Este punto se centra en documentar y justificar las decisiones arquitectónicas tomadas para el proyecto.

Es necesario que utilicen gráficos o lo que tengan al alcance para que sea de fácil comprensión, acompañado de texto.

Arquitectura del sistema:

- Identificar la arquitectura utilizada (por ejemplo, **monolítica, microservicios, cliente-servidor, MVC**, etc.).
- Elaborar un **diagrama de arquitectura del sistema**, explicando cómo interactúan los distintos componentes (base de datos, API, interfaz de usuario, etc.).
- Justificar por qué eligieron esa arquitectura en relación con los requisitos del proyecto, los recursos disponibles y la escalabilidad.

Diseño de bases de datos:

- Presentar el modelo de base de datos utilizado (**Diagrama Entidad-Relación**).
- Explicar las decisiones tomadas en el diseño del esquema de la base de datos (por ejemplo, normalización, índices, claves primarias y foráneas).
- Justificar si eligieron una base de datos relacional (SQL) o no relacional (NoSQL), y por qué.

API rest

Punto 9: Patrones de diseño

En este apartado, se evaluará la comprensión de los patrones de diseño estudiados.

Documentar si aplicaron algún **patrón de diseño** durante el desarrollo (por ejemplo, Singleton, Factory, Observer, etc.).

- Justificar el uso de cada patrón, explicando:
 - Qué problema resuelve.
 - Por qué fue necesario en el proyecto.
 - Cómo se implementó.
 - Incluir un diagrama UML que ilustre cómo se aplicó cada patrón dentro del sistema (las clases involucradas, dependencias etc etc).