**Página principal**

Completar esta sección respondiendo a las siguientes consignas:

* Nombre del Proyecto: DigitalBooking
* Un resumen general de qué consiste el proyecto:  
  El proyecto de reservas de hoteles de alojamiento es un sistema de gestión de reservas de hoteles, departamentos y casas en línea. Este sistema permitirá a los clientes reservar de forma rápida y fácil a través de una interfaz de usuario intuitiva y amigable. Además, el sistema permitirá a los gerentes de hotel [Administrador] gestionar las reservas de forma eficiente, lo que permitirá una mejor atención al cliente y una mayor eficiencia en la gestión del hotel.
* Objetivo que persigue el grupo al realizar el proyecto:  
  El equipo tiene como objetivo crear un sistema de reservas de hotel de alta calidad que proporcione a los clientes una experiencia de reserva de hotel fluida y sin problemas. Además, el grupo busca mejorar la eficiencia de la gestión del hotel mediante la automatización de procesos y la integración de herramientas de análisis de datos. También se espera que el sistema ayude a aumentar la tasa de ocupación de los hoteles, departamentos y casas que ofrece DigitalBooking y a mejorar la satisfacción del cliente. En general, el objetivo principal del grupo es proporcionar una solución completa y efectiva para la gestión de reservas de hoteles de alojamiento.

**Equipo**

Completar esta sección respondiendo las siguientes consignas:

* Listado de cada uno de los integrantes con una breve biografía y descripción de sus habilidades.

**Tomas Alberdi; Desarrolador FrontEnd**

**Tomas Dopaso: Analista de Infraestructura**

**Santiago Ortigoza: Analista de control de calidad/ QA**

**Jonatan Bordon: Desarrollador BackEnd**

Conocimientos sólidos de programación en Java y en el framework SpringBoot.

Capacidad para escribir código limpio, bien estructurado y fácil de mantener.

Experiencia en la creación de servicios RESTful y en el manejo de peticiones HTTP.

Conocimientos básicos de bases de datos relacionales y experiencia en el uso de ORM como Hibernate.

Familiaridad con herramientas de gestión de dependencias como Maven o Gradle.

Conocimiento de patrones de diseño y principios SOLID de programación orientada a objetos.

Capacidad para trabajar con repositorios de código como Git y manejar ramas y conflictos.

Conocimiento de seguridad informática y buenas prácticas de seguridad en el desarrollo de software.

Habilidad para investigar y resolver problemas técnicos.

**Silvestre Novaro: Analista de Base de Datos**

Conocimientos sólidos de SQL y de la estructura de bases de datos relacionales.

Experiencia en el diseño y modelado de bases de datos.

Familiaridad con herramientas de administración de bases de datos como MySQL Workbench.

Conocimientos básicos de programación en lenguajes como Python o Java.

Capacidad para trabajar con conjuntos de datos grandes y complejos.

Conocimiento de técnicas de optimización de consultas y de rendimiento de bases de datos.

Habilidad para escribir scripts de automatización de tareas relacionadas con la base de datos.

Capacidad para trabajar con repositorios de código como Git y manejar ramas y conflictos.

Experiencia en el análisis de datos y en la generación de informes y estadísticas.

Habilidad para investigar y resolver problemas técnicos relacionados con la base de datos.

**Metodología de trabajo**

Completar esta sección respondiendo a las siguientes consignas:

* Breve explicación de la metodología de trabajo utilizada (Scrum), indicando los puntos que les hayan resultado positivos y / o negativos.

En Scrum, el trabajo se divide en sprints de corta duración, que generalmente duran de 2 semanas, donde el equipo se enfoca en entregar un conjunto de funcionalidades acordadas al principio del sprint. Al final de cada sprint, se lleva a cabo una revisión de lo logrado y se planifica el siguiente sprint.

En nuestra opinión, algunos de los puntos positivos de Scrum son que nos ha permitido trabajar en entregables específicos y concretos en un período corto de tiempo. Esto nos permite enfocarnos en tareas importantes y completarlas con éxito dentro del plazo establecido. También encontramos útiles las reuniones diarias (daily scrum) porque nos mantienen actualizados sobre el progreso del proyecto, los obstáculos y los planes para el día siguiente.

Sin embargo, uno de los puntos negativos que hemos experimentado es que a veces puede ser difícil estimar correctamente el tiempo que tomará realizar ciertas tareas. Esto puede resultar en una presión adicional para cumplir con los plazos establecidos pese a esto logramos que no afecte la calidad del trabajo.

En general, creemos que Scrum es una metodología de trabajo eficaz para el equipo de desarrollo, pero como con cualquier proceso, puede haber desafíos y puntos negativos a tener en cuenta. Consideramos que es importante que el equipo se comunique bien y colabore estrechamente para enfrentar estos desafíos y garantizar el éxito del proyecto.

* Explicación de cómo manejaron la asignación de roles dentro del equipo (si los hubo) y cómo se organizaron a la hora de repartir las tareas.

Como equipo de desarrollo, nos reunimos para discutir nuestras fortalezas y debilidades, y cómo podríamos utilizar nuestras habilidades para el éxito del proyecto. En base a estas discusiones, se asignaron roles específicos dentro del equipo, tales como desarrollador frontend [Tomas Alberdi], desarrollador backend [Jonatan Bordon], analista de bases de datos [Silvestre Novaro], analista de infraestructura [Tomas Dopaso] y Analista de control de calidad/ QA o testing analyst [Santiago Ortigoza].

Una vez que se asignaron los roles, discutimos las tareas específicas necesarias para completar el proyecto y repartimos entre los miembros del equipo según nuestras habilidades y experiencias. Para hacerlo, utilizamos un software de gestión de proyectos GitLab [propuesto por DH] que nos permitió crear tareas y asignarlas a diferentes miembros del equipo. Este software nos permitió realizar un seguimiento del progreso de cada tarea y asegurarnos de que no hubiera solapamiento en las tareas asignadas.

También utilizamos una metodología de trabajo ágil Scrum, lo que significa que trabajamos en sprints de corta duración donde el equipo se enfoca en entregar un conjunto de funcionalidades acordadas al principio del sprint. Al final de cada sprint, revisamos lo logrado y planificamos el siguiente sprint.

En general, creemos que la asignación de roles y tareas se manejó bien dentro del equipo. Nos permitió aprovechar las habilidades y fortalezas de cada miembro del equipo para el éxito del proyecto. Además, utilizar un software de gestión de proyectos nos ayudó a trabajar de manera eficiente y asegurarnos de que todos estaban al tanto de lo que se esperaba de ellos.

* Opcionalmente pueden describir otros acuerdos a los que hayan llegado para trabajar en equipo, gestionar el control de versiones, normalizar el código producido, etc.

Como equipo, hemos llegado a algunos acuerdos adicionales para trabajar en equipo de manera efectiva. Estos incluyen:

Gestión del control de versiones: acordamos utilizar GitLab y GitHub como herramienta para controlar el versionado del código fuente. Además, hemos establecido un flujo de trabajo para el control de versiones en el que los miembros de todo el equipo trabajamos en ramas separadas y luego fusionamos nuestro trabajo en la rama principal del repositorio. Esto nos permitió trabajar de manera independiente y colaborar de manera efectiva en el mismo proyecto.

Normalización del código: Acordamos seguir unas convenciones de codificación para garantizar que el código que producimos sea legible y coherente. Esto incluye establecer reglas para el formato del código, nombres de variables, comentarios y otros aspectos relacionados con el estilo del código.

Pruebas de unidad: Acordamos escribir pruebas de unidad para todas las funcionalidades que desarrollamos. Esto nos permitió detectar y corregir errores de manera más eficiente, así como también asegurarnos de que las funcionalidades se comporten como se esperamos.

Revisión de código: Acordamos realizar revisiones de código [por lo general, diariamente] para garantizar que el código producido por todo el equipo cumpla con los estándares acordados y sea de alta calidad. Esto nos permite detectar problemas y errores en una etapa temprana y asegurarnos de que el código producido sea coherente y legible.

En general, estos acuerdos adicionales nos han ayudado a trabajar de manera más efectiva como equipo, garantizando que el código producido sea de alta calidad y que los procesos de desarrollo sean eficientes y coherentes.

**Bitácora del proyecto**

Completar esta sección respondiendo a las siguientes consignas:

* Describir cómo vivieron cada sprint, qué desafíos se encontraron y cómo resolvieron los requerimientos obligatorios y opcionales.
  + Pueden utilizar y referenciar el resultado de las dailies y las retros para este punto.

Como equipo de desarrollo, cada sprint fue un desafío y una oportunidad para seguir mejorando nuestro trabajo. Cada sprint comenzó con una planificación donde definimos los objetivos y las tareas a completar en el sprint.

Durante las dailies, tuvimos la oportunidad de compartir nuestro progreso y abordar cualquier problema o bloqueo que surgió en el camino. Esto nos permitió mantenernos al día con los avances de los demás y trabajar juntos para resolver cualquier problema que pudiera surgir.

Uno de los mayores desafíos que enfrentamos fue asegurarnos de que los requerimientos obligatorios y opcionales fueran abordados de manera eficiente. Para hacer frente a este desafío, trabajamos juntos para establecer prioridades y dividir el trabajo según la capacidad del equipo. También nos aseguramos de tener un enfoque iterativo en el desarrollo [aplicando todo lo aprendido en taller1\_Metodologias del Trabajo], lo que nos permitió ajustar y cambiar nuestro enfoque a medida que surgían nuevas necesidades y requisitos.

Durante las retrospectivas, tuvimos la oportunidad de reflexionar sobre nuestro desempeño en el sprint y encontrar maneras de mejorar en el futuro. Tomamos en cuenta las retroalimentaciones del ProductOwner en este caso nuestro Cliente y las experiencias personales para encontrar áreas de mejora en nuestros procesos y en nuestro trabajo en equipo.

En general, cada sprint fue una oportunidad para seguir mejorando nuestro trabajo y nuestro enfoque en el cumplimiento de los objetivos y requerimientos del proyecto. Trabajando juntos, logramos superar los desafíos y asegurarnos de que todos los requerimientos se abordaran de manera eficiente y efectiva.

* Resumir las de fortalezas del equipo y cómo supieron aprovecharlas para avanzar de mejor manera.
  + Contar puntualmente algo que consideren que haya sido un éxito dentro del equipo / desarrollo.

Como equipo de desarrollo, consideramos que una de las principales fortalezas que hemos tenido ha sido la comunicación efectiva. Desde el comienzo del proyecto, establecimos una comunicación abierta y constante para asegurarnos de que todos estuviéramos en la misma página y para abordar cualquier problema o desafío que surgiera en el camino. Esto para nosotros fue de suma importancia ya que trabajamos en diferentes áreas del proyecto y en diferentes ubicaciones geográficas [Argentina, Colombia].

Otra fortaleza que logramos como equipo es la colaboración, dispuestos a ayudar y trabajar juntos para lograr los objetivos del proyecto. Siempre con una actitud positiva y enfocados en soluciones, esto nos ayudó a lograr la mayor eficiencia en el desarrollo y a la resolución rápida de problemas.

Coincidimos que, uno de los mayores éxitos del equipo ha sido la entrega de un módulo crítico del proyecto cumpliendo los plazos establecidos, intentando siempre buscar una calidad óptima. Aprovechamos nuestras fortalezas en comunicación y colaboración para trabajar de manera eficiente y efectiva en el módulo, abordando cualquier problema de manera oportuna y manteniéndonos enfocados en los objetivos del proyecto.

* Resumen de debilidades del equipo y cómo resolvieron situaciones difíciles o conflictivas.
  + Contar puntualmente algo que consideren que haya salido mal, cómo lo resolvieron y qué aprendieron como consecuencia.

Como equipo en nuestro primer proyecto, hemos enfrentado varias debilidades y situaciones difíciles. Uno de los mayores desafíos fue la falta de experiencia y conocimientos técnicos en ciertas áreas. Esto resultó en errores, pero no así en retrasos en el desarrollo, lo que no afectó la calidad y el tiempo de entrega siendo para nosotros muy importante respecto a las funcionalidades solicitadas en cada sprint.

Para abordar esta situación, nos aseguramos de trabajar en equipo y compartir nuestros conocimientos y habilidades para cubrir las brechas de conocimiento. También nos aseguramos de buscar ayuda externa [compañeros de la camada] y consultar con otros desarrolladores más experimentados como los TL cuando nos encontramos con problemas técnicos.

En un momento específico, nos encontramos con un problema de integración, en lugar de enfocarnos en culpar a alguien por el problema, nos pusimos a trabajar juntos para encontrar una solución. Hicimos una lluvia de ideas sobre posibles soluciones, realizamos pruebas y probamos varias opciones antes de encontrar la solución adecuada. Aprendimos que trabajar juntos en lugar de señalar con el dedo a alguien fue la mejor manera de abordar el problema.

En general, aprendimos que trabajar juntos como un equipo y comunicarnos claramente son claves para superar nuestras debilidades y resolver situaciones difíciles en un proyecto. Es importante aprender de nuestros errores y tomar las medidas necesarias para mejorar continuamente como equipo y como individuos.

**Tecnologías utilizadas**

Completar esta sección respondiendo a las siguientes consignas:

* Qué servicios / herramientas emplearon a la hora de realizar el desarrollo.
  + Gestión del proyecto.

Como equipo de desarrollo en nuestro primer proyecto como desarrolladores, empleamos varias herramientas para la gestión del proyecto. Algunas de estas herramientas incluyen:

GitLab-Boards: Utilizamos esta herramienta para crear y administrar nuestro tablero de proyecto. En el tablero, definimos todas las tareas necesarias para completar el proyecto y las dividimos en diferentes columnas según su estado, como "Open", "En progreso”, “A verificar", "Bug/Error" y "Closed". Esto nos permitió tener una visión clara de lo que se necesitábamos hacer y mantenernos organizados durante todo el proceso.

GitHub y GitLab: Utilizamos GitHub y GitLab como nuestra plataforma de control de versiones y colaboración. Cada miembro del equipo creó su propia rama y cuando se hizo un cambio en el código, se hizo un pull request para revisar y fusionar el código en el repositorio principal [main]. Esto nos permitió colaborar de manera efectiva en el código y mantener un registro completo de todos los cambios realizados.

Google Drive: utilizamos Google Drive para compartir y colaborar en documentos importantes, como la documentación del proyecto y las especificaciones de requisitos.

En general, estas herramientas nos permitieron gestionar de manera efectiva nuestro proyecto y mantenernos organizados durante todo el proceso. Como equipo encontramos estas herramientas accesibles y fáciles de usar para nuestro proyecto.

* + Desarrollo del proyecto:
    - IDEs
    - Sistemas de control de versiones
    - Simulación / Virtualización

Como equipo de desarrollo en nuestro primer proyecto juntos como Jrs. En cuanto al desarrollo del proyecto, hemos estado utilizando diferentes herramientas y tecnologías para llevar a cabo el trabajo de manera efectiva.

En cuanto a las IDEs, hemos estado utilizando principalmente IntelliJ IDEA, popular para el desarrollo en Java con una gran cantidad de herramientas y funcionalidades para mejorar nuestra productividad en el desarrollo.

En cuanto a los sistemas de control de versiones, hemos estado utilizando Git junto con la plataforma de alojamiento de repositorios GitHub/GitLab. Esto nos ha permitido colaborar de manera efectiva en el código y mantener un registro claro de los cambios realizados en el proyecto. Además, GitLab nos ha permitido revisar y aprobar el código de los demás miembros del equipo antes de ser fusionado con el código principal.

En cuanto a la simulación y virtualización, utilizamos diferentes herramientas para probar nuestro código en diferentes entornos. Hemos utilizado Docker para crear contenedores y simulaciones de diferentes sistemas operativos y entornos de desarrollo, lo que nos ha permitido probar y depurar el código en diferentes configuraciones sin tener que configurar físicamente cada entorno. En cuanto a herramientas de virtualización tenemos conocimiento en VirtualBox para crear máquinas virtuales y simular diferentes entornos de hardware y software, pero no nos vimos en la necesidad de implementarlo.

En general, estas herramientas y tecnologías nos han ayudado mucho en el desarrollo del proyecto y nos han permitido colaborar de manera efectiva como equipo de desarrollo junior.

* Qué tecnologías utilizaron para el desarrollo del sitio

Front end: HTML5, CSS3, JavaScript, React

Back end: Java, Spring Boot, REST API

Base de datos: MySQL

Infraestructura: Amazon Web Services (AWS), Git

Testing: JUnit, Mockito, Vitest \*investigar, ReactTestingLibrary

**Documentación técnica del proyecto**

Esta sección la dejamos a su criterio. Básicamente debería contener toda la información técnica que pueda serle útil a los integrantes del equipo o personas de deban comprender el funcionamiento del sitio desde el lado técnico.

Algunas sugerencias:

* Ambiente de desarrollo
  + Instrucciones de cómo un nuevo integrante debería armarse su ambiente de desarrollo

Para poder armar un ambiente de desarrollo similar al del equipo se debe seguir los siguientes pasos:

**Desarrollador BackEnd:**

Descargar e instalar la versión Community de IntelliJ IDEA (actualmente la versión 2022.2.3).

Descargar e instalar MySQL (puede ser la versión comunitaria o cualquier otra versión que soporte la conexión JDBC).

Clonar el repositorio del proyecto desde la plataforma de control de versiones que se esté utilizando (por ejemplo, GitLab, GitHub, etc.).

Abrir IntelliJ IDEA e importar el proyecto desde el directorio donde se clonó el repositorio.

--------------------------------------------------------------------------------------------------------

Configurar las siguientes dependencias del proyecto en el archivo pom.xml :

org.springframework.boot:spring-boot-starter-data-jpa

org.springframework.boot:spring-boot-starter-validation

org.springframework.boot:spring-boot-starter-web

com.mysql:mysql-connector-j

org.projectlombok:lombok (opcional)

org.springframework.boot:spring-boot-starter-test (scope: test)

com.fasterxml.jackson.core:jackson-databind (version: 2.13.1)

org.springframework.boot:spring-boot-starter-mail

org.springframework.boot:spring-boot-starter-security

org.springframework.security:spring-security-crypto (version: 6.0.2)

org.springframework.security:spring-security-oauth2-resource-server (version: 6.0.2)

io.jsonwebtoken:jjwt-api (version: 0.11.5)

io.jsonwebtoken:jjwt-impl (version: 0.11.5, scope: runtime)

io.jsonwebtoken:jjwt-jackson (version: 0.11.5, scope: runtime)

org.springdoc:springdoc-openapi-starter-webmvc-ui (version: 2.0.0)

--------------------------------------------------------------------------------------------------------

Configurar las propiedades de conexión a la base de datos en el archivo application.properties según :

Especifica la URL de la base de datos, que incluye el protocolo JDBC, el servidor y la base de datos a la que se conectará la aplicación.

**spring.datasource.url=jdbc:mysql://digitalhouse.cluster-cnyqegjgetrv.us-east-2.rds.amazonaws.com/0223TDPROM1C2LAED0522FT\_GRUPO7?useSSL=true&verifyServerCertificate=false**

--------------------------------------------------------------------------------------------------------

Nombre de Usuario y la Contraseña para acceder a la base de datos.

**spring.datasource.username=0223TDPROM1C2LAED0522FT\_GRUPO7**

**spring.datasource.password=hae9Eibu**

--------------------------------------------------------------------------------------------------------

Establece el dialecto de Hibernate para MySQL y especifica que Hibernate debe actualizar automáticamente la estructura de la base de datos a medida que se realizan cambios en las entidades de la aplicación.

**# Hibernate properties**

**spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect**

**spring.jpa.hibernate.ddl-auto=update**

--------------------------------------------------------------------------------------------------------

Configurar cualquier otra propiedad del proyecto que sea necesaria para su correcto funcionamiento (por ejemplo, las credenciales para el envío de correos electrónicos).

Ejecutar el proyecto y verificar que se puedan realizar las operaciones básicas (como registrar un usuario y obtener un token de autenticación) mediante la interfaz Swagger.

Además de estas instrucciones, es importante que el nuevo integrante tenga un conocimiento básico de Java, Spring Boot, JPA, MySQL y Swagger para poder entender el código y realizar las tareas asignadas en el proyecto.

--------------------------------------------------------------------------------------------------------

**Desarrollador Analista de Datos:**

Como Analista de datos en MySQL Workbench, para armar tu ambiente de desarrollo, debes seguir los siguientes pasos:

Descarga e instala MySQL Workbench en tu equipo. Puedes descargarlo desde el sitio oficial de MySQL.

Configura una conexión a la base de datos. Para hacerlo, abre MySQL Workbench y selecciona la opción "New Connection" en la pantalla de bienvenida. En la ventana de configuración, completa la información requerida, incluyendo el host, puerto, nombre de usuario y contraseña. En este caso, deberás utilizar la información de conexión proporcionada en las propiedades de BackEnd:

**Host: digitalhouse.cluster-cnyqegjgetrv.us-east-2.rds.amazonaws.com**

**Puerto: 3306 (por defecto para MySQL)**

**Nombre de usuario: 0223TDPROM1C2LAED0522FT\_GRUPO7**

**Contraseña: hae9Eibu**

Una vez que hayas configurado la conexión, podrás conectarte a la base de datos haciendo clic en el botón "Test Connection". Si la conexión es exitosa, haz clic en "OK" para guardar la configuración.

Con estos pasos, deberías estar listo para empezar a trabajar en el proyecto como Analista de datos en MySQL Workbench.

--------------------------------------------------------------------------------------------------------

**Desarrollador Desarrollador FronEnd:**

Como desarrollador FrontEnd, para poder trabajar en el proyecto necesitarás tener instalado lo siguiente en tu ambiente de desarrollo:

Node.js: Lo primero que necesitarás es tener instalado Node.js en tu sistema. Puedes descargar la versión más reciente desde el sitio web oficial de Node.js.

Visual Studio Code: Este es un editor de código gratuito y muy popular entre desarrolladores. Puedes descargarlo desde su sitio web oficial.

Git: Git es un sistema de control de versiones que utilizamos para colaborar en el proyecto. Si aún no lo tienes instalado, puedes descargarlo desde su sitio web oficial.

NPM: NPM es el gestor de paquetes de Node.js que utilizamos para instalar las dependencias del proyecto. Se instala automáticamente cuando instalas Node.js.

Una vez que hayas instalado estas herramientas, debes seguir estos pasos:

Clona el repositorio del proyecto desde Github en tu equipo utilizando Git.

Abre una terminal de línea de comandos y navega hasta el directorio del proyecto clonado.

Ejecuta el comando npm install para instalar todas las dependencias del proyecto.

React Router Dom: Es la librería que utilizamos para manejar el enrutamiento en la aplicación. Para instalarla, ejecuta el siguiente comando en la terminal:

**npm install react-router-dom**

Fslightbox React: Es la librería que utilizamos para implementar la galería de fotos en la versión de escritorio. Para instalarla, ejecuta el siguiente comando en la terminal:

npm install fslightbox-react

Swiper: Es la librería que utilizamos para implementar el slider de fotos en las versiones móvil y tablet. Para instalarla, ejecuta el siguiente comando en la terminal:

npm install swiper

React Datepicker: Es la librería que utilizamos para implementar el seleccionador de fechas en el filtro y reserva. Para instalarla, ejecuta el siguiente comando en la terminal:

npm install react-datepicker

Reactjs Availability Calendar: Es la librería que utilizamos para implementar el calendario de disponibilidad de productos. Para instalarla, ejecuta el siguiente comando en la terminal:

npm install reactjs-availability-calendar

Una vez que se hayan instalado todas las dependencias y librerias, puedes ejecutar el proyecto utilizando el comando npm run dev.

Ahora puedes acceder al proyecto en tu navegador web en la dirección <http://localhost:3000>.

* + Buenas prácticas / convenciones / acuerdos a la hora de desarrollar

Como equipo de desarrollo, creemos que es importante establecer algunas buenas prácticas, convenciones y acuerdos para asegurarnos de que nuestro proyecto se desarrolle de manera efectiva y coherente. Algunas de las prácticas que consideramos incluyen:

Establecer un patrón de codificación coherente: Para mantener el código limpio y fácil de entender, acordamos un patrón de codificación coherente que incluya convenciones para la nomenclatura de variables, identación, uso de comentarios, etc.

Utilizar control de versiones: El uso de un sistema de control de versiones, como GitHub o GitLab, nos permitirá mantener un registro de los cambios que se realizan en el código, así como revertir los cambios en caso de que sea necesario.

Realizar pruebas de forma regular: A medida que desarrollamos funcionalidades, es importante realizar pruebas de forma regular para asegurarnos de que todo funciona como debería. Podríamos acordar una estrategia de pruebas que incluya pruebas unitarias y de integración.

Comunicación efectiva: Es importante que todos los miembros del equipo estén al tanto de los avances, problemas y próximos pasos. Podríamos establecer reuniones regulares para discutir el progreso y cualquier problema que surja.

Documentación: Para asegurarnos de que todos los miembros del equipo estén al tanto de la estructura del proyecto y cómo funciona, podríamos acordar la documentación del código y las funcionalidades en un wiki o un sistema similar.

* + Buenas prácticas respecto a los commits, merges y el uso de branches

Como equipo en nuestro primer proyecto, es importante establecer buenas prácticas en cuanto a los commits, merges y el uso de branches. Aquí detallamos algunas implementaciones:

Commits significativos: Los commits deben ser significativos y deberían incluir solo los cambios relacionados con una tarea específica o una funcionalidad. Esto nos va ayudar a rastrear los cambios de manera efectiva y nos permitirá revertir los cambios si es necesario.

Mensajes claros de commit: Los mensajes de commit deben ser claros y concisos para que cualquier persona pueda entender fácilmente el propósito del commit.

Uso de ramas (branches): Debemos utilizar ramas para separar el trabajo en progreso del código que ya ha sido aprobado y se encuentra en producción. Es importante nombrar nuestras ramas de manera clara y concisa para que otros miembros del equipo puedan entender el propósito de cada rama.

Code reviews: Realizar revisiones de código (code reviews) antes de realizar un merge es importante para asegurarnos de que el código sea de alta calidad y siga las mejores prácticas. También nos permite detectar posibles errores o problemas antes de que lleguen a producción.

Merge con cuidado: Al realizar un merge, debemos asegurarnos de que todos los conflictos se hayan resuelto de manera efectiva y de que se hayan realizado las pruebas necesarias para confirmar que el código se integra correctamente.

De esta manera podemos asegurarnos de que nuestro código sea de alta calidad y fácil de mantener. También nos permitirá colaborar de manera efectiva como equipo, reducir los errores y mantener una historia clara de los cambios realizados en el proyecto.

* Base de datos
  + Estructura y diagramas
  + Scripts / instrucciones para la creación de la base de datos
  + Scripts / set de datos dummy
* Componentes y diagramas de la aplicación
  + Principales partes de la aplicación y cómo se conectan
  + APIs disponibles y su documentación (pueden linkear a un sitio externo)
* Infraestructura
  + Estructura y diagramas
* Testing y calidad
  + Incluir el informe requerido para el cuarto sprint y cualquier otra información referente a cómo buscaron asegurar la calidad del software producido