Estructura del Repositorio

El repositorio en GitHub está diseñado como un espacio central donde se almacena todo el contenido del proyecto. Su organización se basa en carpetas específicas que separan las diferentes partes del sistema, facilitando la colaboración y el acceso a los recursos. A continuación, se describe su estructura:

docs/: Contiene la documentación del proyecto, como guías, manuales o cualquier material explicativo que detalle el funcionamiento o el propósito del sistema.

src/: Alberga el código fuente, dividido en subcarpetas según su función:

frontend/: Incluye los elementos de la interfaz de usuario, como archivos HTML, CSS o JavaScript, si el proyecto tiene una parte visual.

backend/: Agrupa la lógica del sistema, como scripts en Python, Java u otros lenguajes que gestionen las funcionalidades internas.

database/: Reúne los archivos relacionados con la base de datos, como scripts SQL o modelos de datos.

resources/: Destinada a los recursos estáticos, como libros en formato PDF, imágenes u otros archivos de apoyo al aprendizaje.

tests/: Espacio reservado para pruebas del código, en caso de que se implementen.

README.md: Archivo inicial que describe el proyecto, su propósito y los pasos básicos para entenderlo o empezar a trabajar en él.

.gitignore: Archivo que especifica qué elementos (como archivos temporales o carpetas generadas automáticamente) no deben incluirse en el repositorio.

Esta organización permite que cada componente del proyecto tenga un lugar claro, lo que simplifica la colaboración entre los integrantes.

Estructura de Ramas

El trabajo en el repositorio se gestiona mediante un sistema de ramas,lo cual permite a cada persona desarrollar sus tareas de forma independiente sin afectar el proyecto principal. Las ramas están estructuradas de la siguiente manera:

main: Representa la versión estable y definitiva del proyecto. Solo contiene cambios completamente probados y aprobados.

dev: Actúa como una rama intermedia de desarrollo, donde se integran los aportes de todos antes de pasar a main. Es el punto de encuentro para los cambios en progreso.

Ramas de características (feature/): Son ramas temporales creadas por cada integrante para trabajar en tareas específicas. Los nombres reflejan el propósito de la tarea, como feature/frontendAngieo feature/diseño-interfaz.

De esta forma, el proyecto avanza en paralelo, con cada persona enfocada en su propia rama, mientras dev sirve como un espacio de integración y main mantiene la versión final.

Flujo de Trabajo

El desarrollo colaborativo sigue un flujo estructurado que asegura que los cambios se realicen de manera ordenada y se integren sin conflictos. Este proceso se basa en el modelo GitHub Flow y se describe a continuación:

Inicio del trabajo: Todo comienza desde la rama dev, que contiene la versión más actualizada del proyecto en desarrollo. Cada integrante parte de esta rama para crear su propia rama de características.

Desarrollo individual: En su rama personal (por ejemplo, feature/frontendAngie, cada persona realiza los cambios necesarios, como agregar código, subir recursos o editar documentación.

Registro de cambios: Los avances se guardan localmente y luego se suben al repositorio en GitHub como una rama independiente.

Integración: Una vez completada la tarea, se solicita combinar la rama personal con dev a través de un Pull Request (PR) en GitHub. Este PR incluye una descripción de los cambios y permite revisarlos antes de su aprobación.

Revisión y ajuste: Los PRs son revisados por mi. Si hay conflictos o ajustes necesarios, se resuelven antes de la integración.

Actualización de main: Cuando dev alcanza un estado estable y funcional, sus cambios se combinan con main para actualizar la versión oficial del proyecto.

Este flujo garantiza que el trabajo de cada integrante se incorpore de manera controlada, manteniendo la estabilidad del sistema.

Distribución del Equipo

Con 9 personas involucradas, el proyecto puede dividirse en áreas específicas para optimizar el esfuerzo. Una posible distribución incluye:

Desarrollo de la interfaz (frontend/): 2 personas.

Implementación de la lógica del sistema (backend/): 2 personas.

Gestión de la base de datos (database/): 2 personas.

Carga y organización de recursos (resources/): 2 personas.

Coordinación y revisión general: 1 persona.

Esta división permite que cada subgrupo trabaje en paralelo, enfocándose en su parte del proyecto, mientras las ramas individuales evitan interferencias.

Beneficios de la Estructura

El uso de esta organización y flujo de trabajo ofrece varias ventajas:

Orden: Las carpetas y ramas mantienen los archivos y cambios bien organizados.

Colaboración: Cada persona tiene su propio espacio para trabajar sin afectar a los demás.

Control: Los Pull Requests permiten revisar y aprobar los cambios antes de integrarlos.

Estabilidad: La separación entre main y dev asegura que la versión final siempre sea funcional.

Comandos Importantes de Git

A continuación, se describen algunos comandos clave de Git que se utilizan en este flujo de trabajo:

git clone https://github.com/axchisan/El-rincon-de-ADSO.git Descarga una copia del repositorio desde GitHub a la computadora local.

git checkout <nombre-rama>: Cambia a una rama específica, como dev o una rama personal.

git checkout -b <nombre-rama>: Crea y cambia a una nueva rama (por ejemplo, feature/agregar-libros).

git add .: Prepara todos los cambios realizados para ser guardados.

git commit -m "Mensaje": Guarda los cambios con una descripción breve de lo que se hizo.

git push origin <nombre-rama>: Sube la rama al repositorio en GitHub.

git pull origin dev : Actualiza la copia local con los cambios más recientes de la rama de desarrollo

Estos comandos son esenciales para interactuar con el repositorio y forman la base del trabajo colaborativo.

Conclusión

La estructura del proyecto para el SENA combina una organización clara de archivos con un sistema de ramas y un flujo de trabajo basado en GitHub Flow. Esto permite que un equipo de 9 personas colabore eficientemente en la creación de un sistema de gestión de recursos de aprendizaje, manteniendo el control sobre los cambios y asegurando una versión estable del proyecto en todo momento. Con esta guía, se busca proporcionar una visión general que facilite la comprensión y participación de todos los involucrados.