Resumen Ejecutivo

El proyecto de la startup “EduTech IA” consta de adaptar el contenido de los cursos escolares a las necesidades individuales de cada alumno utilizando herramientas de inteligencia artificial.

Selección del Lenguaje y Paradigma de Programación

Según la descripción del proyecto, el paradigma más apropiado es la Programación Orientada a Objetos (POO / OOP). Esto se debe a que hay múltiples partes que están relacionadas entre sí, lo que se podría modelar de una forma bastante cómoda utilizando las herramientas que nos proporciona la orientación a objetos.

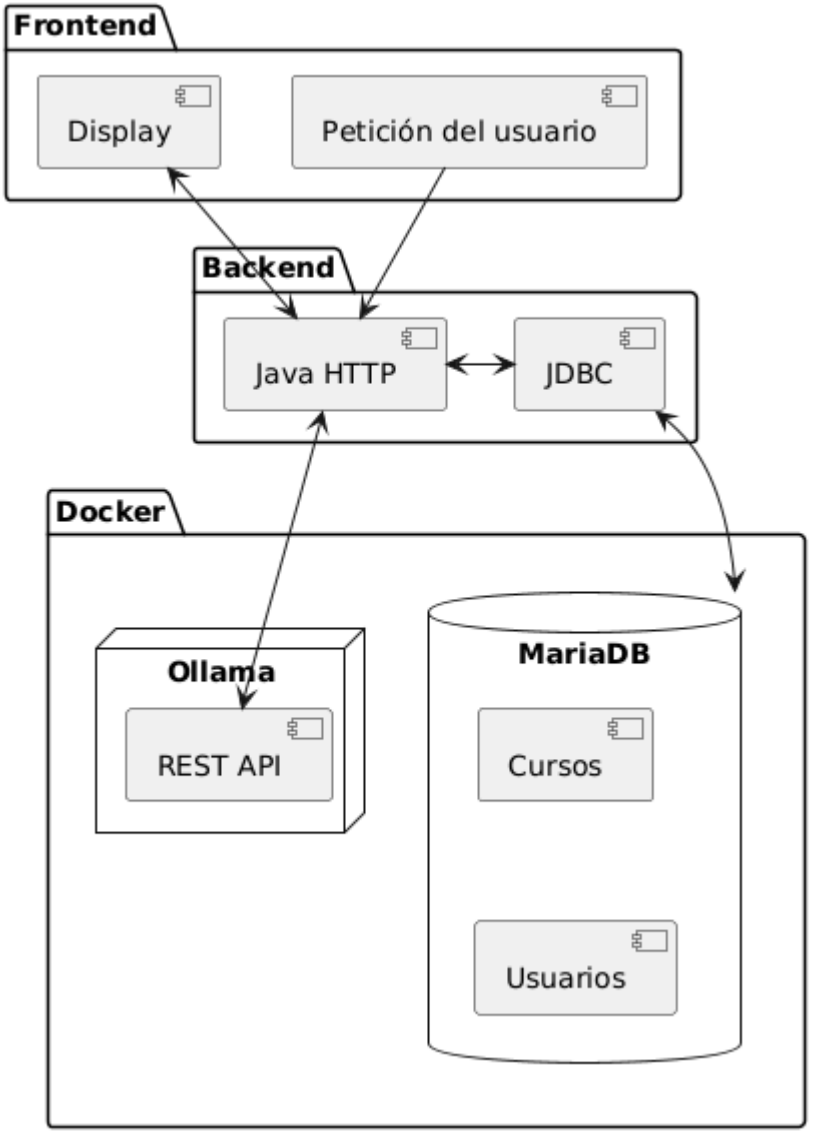
A la hora de escoger un lenguaje de programación, debemos tener en cuenta qué se quiere integrar en el producto final, en este caso, se quiere integrar elementos de IA para ajustar los elementos de los cursos a las necesidades individuales de los alumnos. Para llevar esto a cabo podemos seguir varios caminos. Una opción sería entrenar un modelo específico de IA para ese propósito y utilizarlo, otra sería intentar utilizar un modelo de lenguaje grande (LLM) ya existente como, puede ser LLaMA, y ejecutarlo en una máquina local. Por último, otra opción sería utilizar un LLM externo, como ChatGPT o DeepSeek, a través de su API y así utilizar modelos grandes que no podríamos ejecutar en local.

Para ayudarnos a decidir que escoger, podemos apoyarnos en que uno de los requisitos es la seguridad, esto nos ayudaría a descartar la tercera opción, ya que estaríamos enviando datos de nuestros usuarios y sus cursos a terceros al utilizar sus APIs. Por esto mismo, el lenguaje que recomendaría sería Python, utilizando sus múltiples librerías enfocadas a Machine Learning y sus herramientas para orientación a objetos.

Antes de decidirnos por completo, existe la posibilidad de utilizar múltiples lenguajes, por ejemplo, haciendo la parte de IA con Python y, comunicándose mediante una API REST, enlazarlo con algún otro lenguaje. Así podríamos aprovecharnos de las librerías de Python, pudiendo utilizar otros lenguajes con los que estemos más familiarizados para construir el resto de la aplicación, como pueden los que mencionamos antes, Java, C#, Ruby…
Por otro lado, podemos apoyarnos en herramientas como Ollama para ejecutar el LLM de forma local, ya que nos proporciona una API con la que interactuar con el modelo.

Teniendo todo esto en cuenta, creo que la opción que mejor le viene a este proyecto es utilizar Ollama para ejecutar el LLM que veamos oportuno, utilizando Java para el resto de la aplicación, ya que es un lenguaje bastante conocido y utilizado en multitud de otras aplicaciones.

Diseño de la Arquitectura del Software



En este diagrama se ilustra de forma simplificada el funcionamiento de la aplicación. Tendremos un Frontend, que puede ser una interfaz web que se comunica mediante peticiones HTTP con una API hecha en Java con Spring Boot para poder comunicarse con el resto del entorno.

El Backend está compuesto por la API ya mencionada, que se comunicará con la API de Ollama y, utilizando el driver JDBC de MariaDB, se comunicará con la base de datos.

La base de datos será MariaDB debido a que es open source, en vez de MySQL que es propiedad de Oracle.

Selección de Herramientas y Configuración del Entorno de Desarrollo

Para el entorno de desarrollo del proyecto, se utilizaran las siguientes herramientas:

- Los IDEs IntelliJ IDEA y PyCharm de la suite de JetBrains. Debido a su alta variedad de herramientas y ayudas.
- Git y GitHub para el control de versiones. Debido a la comodidad que ofrece GitHub a la hora de trabajar en equipo.
- Para comunicarnos se utilizará Slack, ya que es una de las herramientas más utilizadas para este propósito.
- Docker, para ejecutar el servidor de Ollama y el de MariaDB en un entorno controlado.

Usaremos Docker para poder tener siempre las mismas condiciones y especificaciones en el entorno, así eliminamos el problema de “en mi máquina funciona”, estandarizando el entorno.

En vez de Slack podríamos utilizar herramientas como Microsoft Teams, considero que Slack es más común a la hora de hacer este tipo de proyectos de desarrollo. Una opción a considerar es Mattermost, que es una alternativa open source a Slack y Microsoft Teams que cuenta con integraciones con múltiples servicios de CI/CD y plataformas como GitHub, GitLab, CircleCI, Jenkins o Bitbucket.

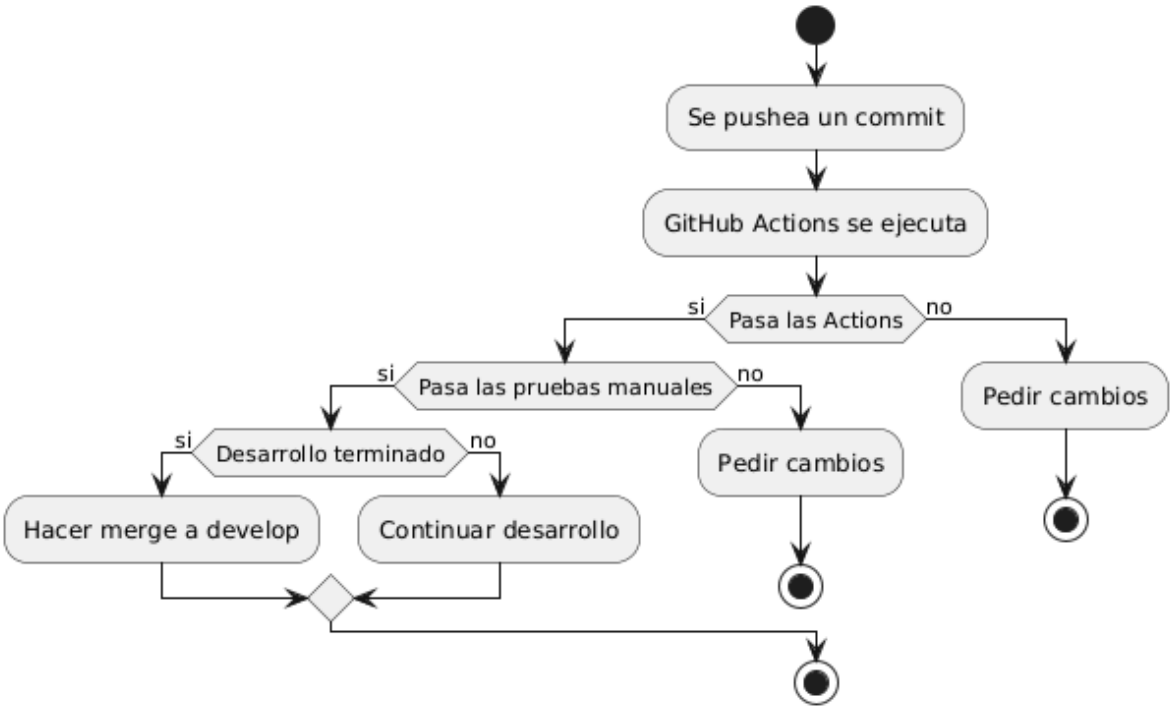
Si se quisiera utilizar un IDE alternativo, se podría utilizar VSCodium, una build de VSCode compilada desde el repositorio oficial, pero sin la telemetría y recolección de datos de Microsoft.

Por otro lado, podríamos utilizar GitLab en vez de GitHub y, junto con las alternativas open source mencionadas anteriormente tendríamos todo el entorno configurado con herramientas open source.

Planificación del Pipeline de Integración y Entrega Continua (CI/CD)

Para el CI/CD utilizaremos GitHub Actions. El pipeline será el siguiente:

1. Al pushear un commit, se ejecutarán las GitHub Actions, que verificarán si el código pasa los tests preparados anteriormente.
    a. Las Actions se ejecutarán dentro de una instancia de Ubuntu.
2. Si pasan los tests, se probarán manualmente aquellas cosas que no puedan ser verificadas fácilmente mediante tests.
3.
    a. Si no pasa alguno de los dos pasos anteriores, se pedirá una modificación o rectificación.
    b. Si ha pasado todas las pruebas anteriores y está terminado, se mergeará a la branch apropiada en caso de ser necesario.



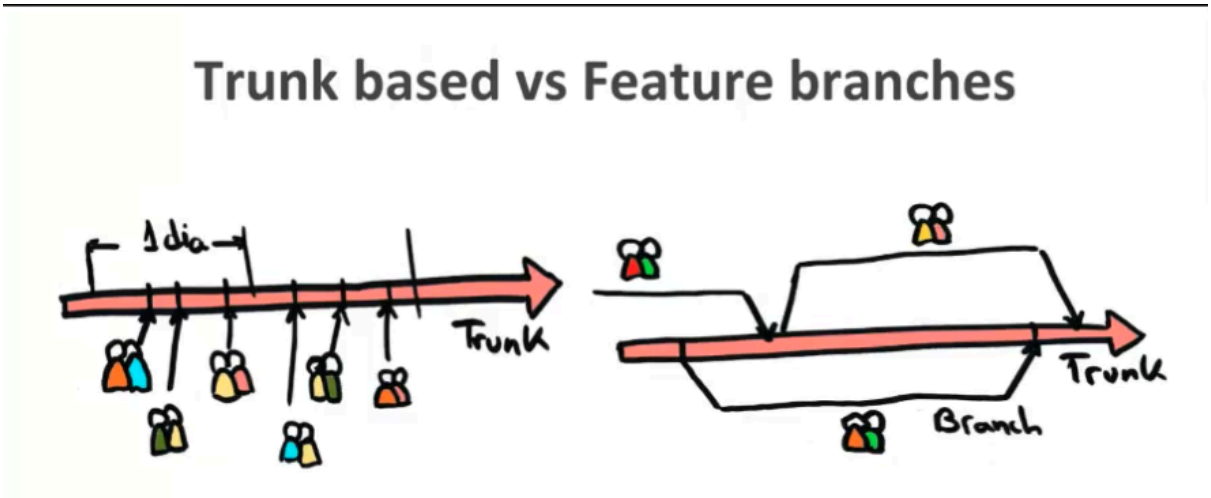
Desarrollo de un Protocolo de Seguridad en el Desarrollo

Vamos a tener 3 ramas principales: *master*, donde estará la versión de producción, *develop*, donde trabajaremos y subiremos los cambios y *testing*, donde estarán las versiones antes de ir a producción.

Además, trabajaremos utilizando feature branches y hotfixes cuando sea necesario.

Los cambios a develop deberán ir mediante un pull request para poder ser verificados, se deberá de esperar a la verificación de otro compañero de equipo antes de hacer merge a develop.

Un ejemplo de cómo será el uso de las ramas es el siguiente:



Fuente: https://raul-profesor.github.io/DEAW/cicd/#trunk-based-vs-feature-branches

Nosotros usaremos la filosofía de las Feature branches en la rama de *develop*.

Estrategia de Gestión y Optimización de Entornos

El entorno de desarrollo estará compuesto por las siguientes herramientas:

- Los IDEs IntelliJ IDEA y PyCharm de la suite de JetBrains. Debido a su alta variedad de herramientas y ayudas.
- Git y GitHub para el control de versiones. Debido a la comodidad que ofrece GitHub a la hora de trabajar en equipo.
- Docker, para ejecutar el servidor de Ollama y el de MariaDB en un entorno controlado.

El entorno de producción será basado en docker, con los siguientes contenedores:

- Ollama (`ollama/ollama`), para el LLM al que se le harán las consultas, con el puerto `11434`, en el que corre Ollama por defecto, expuesto a la máquina host.
- MariaDB (`mariadb:latest`), para la base de datos, con el puerto `3306`, el puerto por defecto de MariaDB, expuesto a la máquina host.
- Ubuntu (`ubuntu:latest`), modificado mediante un DOCKERFILE para instalar java y clonar la rama master cuando haya cambios en ella al arrancar el contenedor.

Conclusiones

Al final, el proyecto requiere de saber manejarse con Docker, debido a su extensivo uso de múltiples contenedores. Se recomienda utilizar el contenedor de Portainer al trabajar en el entorno local para manejar los múltiples contenedores de una forma más cómoda.