Bot de Recomendación de Optativas

Informe Técnico del Proyecto

Desarrolladores:

Adrián Hernández Castellanos - C312

Laura Martir Beltrán - C311

Yesenia Valdés Rodríguez - C311

Gabriel Alonso Coro - C312

Josué Rolando Naranjo Sieiro - C311

1. Resumen del Proyecto

Este proyecto presenta el desarrollo de un sistema inteligente basado en un bot de Telegram, diseñado para asistir a estudiantes universitarios en la elección de asignaturas optativas. A través de una interfaz conversacional intuitiva, los usuarios pueden explorar optativas, calificarlas, dejar reseñas y recibir recomendaciones personalizadas. Por su parte, los profesores cuentan con herramientas para gestionar el sistema de optativas, validar archivos y supervisar la actividad del bot. El sistema hace uso de estructuras de datos en formato JSON, control de autenticación y técnicas de recuperación de información para garantizar una experiencia eficiente y segura.

2. Componentes Principales

La arquitectura del proyecto se compone de varios módulos, cada uno responsable de una funcionalidad específica:
- `main.py`: Implementa la lógica principal del bot, gestionando los flujos conversacionales, los comandos disponibles, la autenticación de profesores, y la interacción con los datos.
- `search_engine.py`: Motor de búsqueda avanzado que combina TF-IDF y técnicas de ponderación para ofrecer resultados relevantes y adaptados a las preferencias del usuario.
- Archivos de datos `data/`: Contienen la información estructurada sobre estudiantes, optativas, profesores y reseñas. Estos archivos actúan como la fuente central del conocimiento del sistema.
Cada componente está diseñado para ser modular, facilitando futuras ampliaciones o adaptaciones del sistema.

3. Autenticación de Profesores

El sistema incorpora un mecanismo de autenticación para profesores mediante el comando /login, que requiere credenciales válidas. Se distingue entre profesores regulares y un perfil especial de superadmin, el cual cuenta con permisos ampliados para modificar datos sensibles y ejecutar acciones administrativas. Una vez autenticado, el profesor mantiene su sesión activa gracias a la identificación por ID de Telegram, lo cual evita accesos no autorizados y facilita un control más preciso de las operaciones realizadas.

4. Manejo de Optativas y Estudiantes

El bot permite a los profesores gestionar completamente el ciclo de vida de las optativas y los estudiantes asignados a ellas. Las operaciones disponibles incluyen:
- Agregar o eliminar registros de estudiantes y profesores.
- Crear nuevas optativas definiendo campos como nombre, profesor responsable, descripción, número de plazas y asignaturas relacionadas.
- Asignar estudiantes a optativas específicas mediante comandos interactivos, lo cual permite mantener una correspondencia clara y trazable entre cada usuario y su elección académica.
Estas acciones están restringidas a usuarios autenticados, lo que garantiza la integridad del sistema.

5. Sistema de Reseñas

Con el objetivo de fomentar la retroalimentación y mejorar las decisiones futuras, el sistema permite que cada estudiante deje una reseña sobre la optativa que tiene asignada. La reseña incluye una puntuación numérica del 1 al 5 y un comentario descriptivo. Si el estudiante ya ha emitido una reseña, la nueva entrada sustituye automáticamente a la anterior. Todas las reseñas se almacenan en el archivo reseñas.json, manteniendo así un registro ordenado y fácilmente accesible. Además, el proceso puede cancelarse en cualquier momento para evitar entradas incompletas o involuntarias.

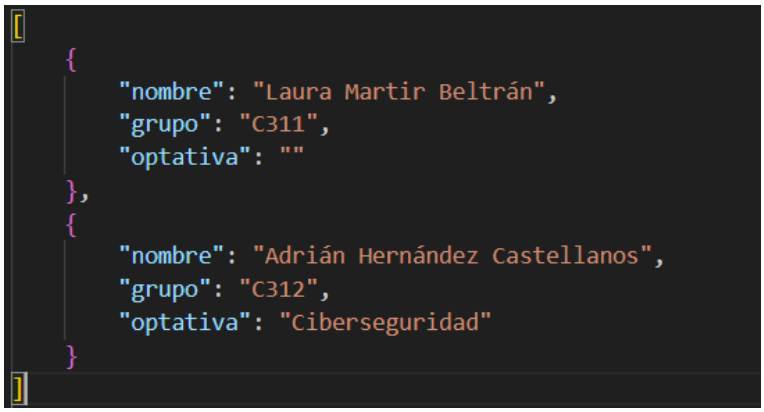
6. Motor de Búsqueda Inteligente

Uno de los elementos más importantes del proyecto es su motor de búsqueda, diseñado para ofrecer recomendaciones y resultados relevantes. Entre sus características destacan:
- Modelo de similitud TF-IDF para evaluar la relevancia de términos en función de su frecuencia y contexto.
- Ponderación personalizada mediante una sintaxis como `palabra***`, que incrementa el peso de ciertos términos en la búsqueda.
- Exclusión de términos mediante el uso de `!palabra`, permitiendo refinar consultas y eliminar resultados no deseados.
- Priorización semántica, favoreciendo optativas con mejores calificaciones y comentarios más positivos.
- Cualquier término insertado en la búsqueda que aparezca en un campo relacionado a una optativa resultará en otorgar importancia a esa optativa a la hora de mostrar una respuesta a la consulta. Los campos incluyen nombre de la optativa, nombre del profesor que la imparte, descripción y asignaturas relacionadas.
Estas funcionalidades permiten realizar consultas complejas de manera natural, acercando al usuario a una experiencia de búsqueda más inteligente y adaptable.

7. Carga de Archivos por Profesores

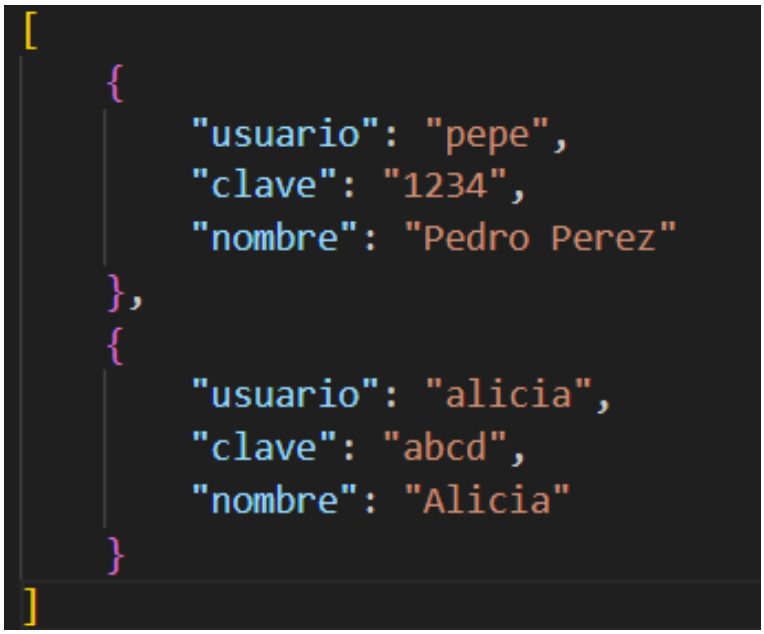
El sistema permite a los profesores autenticados subir archivos .json para actualizar la base de datos del sistema. Solo se admiten archivos con nombres específicos (estudiantes.json, optativas.json, profesores.json) y con estructuras válidas. La codificación de los archivos debe ser UTF-8 sin escape ASCII. El bot valida automáticamente el contenido antes de reemplazar los datos existentes, garantizando así la consistencia y evitando corrupciones accidentales. Esta funcionalidad proporciona una vía rápida y segura para mantener actualizado el sistema sin intervención manual en el servidor. A continuación se muestra el formato esperado de cada archivo .json reemplazable.

estudiantes.json



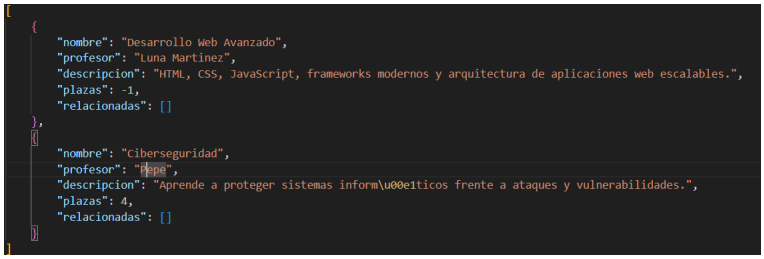
El archivo de estudiantes debe consistir en un archivo en formato lista, donde cada elemento debe contener un apartado nombre, un apartado grupo y un apartado optativa. Excepto optativa, ningún otro apartado debe permanecer vacío. El apartado optativa vacío implica que el estudiante no está asignado a ninguna optativa.

profesores.json



El archivo de profesores debe consistir en un archivo en formato lista, donde cada elemento debe contener un apartado nombre, un apartado usuario y un apartado clave, que representan el nombre real, nombre de usuario a insertar durante el login y contraseña. Ningún apartado debe permanecer vacío.

optativas.json



El archivo de optativas debe consistir en un archivo en formato lista, donde cada elemento debe contener un apartado nombre (no vacío), un apartado profesor (no vacío), un apartado descripción (no vacío), un apartado plazas, en enteros positivos que corresponden a la cantidad de plazas disponibles y un apartado relacionadas, que representa una lista de string de nombres de asignaturas relacionadas, que puede ser una lista vacía en caso en que se requiera. Si el valor de plazas es -1, entonces la optativa se considerará abierta a todos quienes quieran participar, sin límites de plazas.

8. Registro de Operaciones

Para garantizar la trazabilidad y responsabilidad en la gestión del sistema, cada acción realizada por un profesor queda registrada en el archivo `logs/registro_operaciones.txt`. Se almacena información detallada sobre qué acción se realizó, por quién, en qué momento y sobre qué entidad (estudiante, optativa o archivo). Este registro se limita a las últimas 1000 entradas, evitando así el crecimiento descontrolado del archivo. Además, mediante el comando `/log`, los profesores pueden descargar dicho historial para su revisión.

9. Comandos del Bot

El bot responde a una serie de comandos estructurados que permiten una interacción fluida y sencilla tanto para estudiantes como profesores:
- `/start`: Muestra las optativas disponibles al usuario.
- `/help`: Despliega una guía de uso y comandos disponibles. Esta guía cambia si se está logueado como profesor.
- `/rev`: Permite al estudiante escribir una reseña sobre su optativa asignada.
- `/vrev`: Muestra las reseñas realizadas sobre una optativa específica.
- `/login`: Inicia sesión como profesor o superadmin.
- `/log`: Descarga el historial de operaciones recientes (solo para profesores).
- `/delrev`: Elimina todas las reseñas del sistema (solo superadmin).

10. Superadmin

Como fue mencionado anteriormente, el superadmin es un usuario de categoría profesor con permisos administrativos especiales, que puede agregar y eliminar profesores y modificar el archivo profesores.json (es el único usuario que puede ejecutar estas operaciones por motivos de seguridad). Sus acciones quedarán registradas en el registro de la misma manera que las de cualquier otro usuario, y podrá realizar las mismas acciones que estos últimos. Para autenticarse como superadmin, se debe ejecutar el comando /login y se debe insertar como usuario: superadmin y contraseña: admin1234

11. Requisitos del Proyecto

El archivo `requirements.txt` especifica las dependencias necesarias para ejecutar correctamente el proyecto. Las bibliotecas utilizadas se centran en el desarrollo del bot, el procesamiento de lenguaje natural y la recuperación de información:
- `python-telegram-bot==20.8`: Librería principal para la creación y gestión del bot de Telegram.

- - `scikit-learn==1.4.1.post1`: Utilizada para la implementación del motor TF-IDF y procesamiento de texto.
- `numpy>=1.24`: Apoyo en operaciones numéricas y estructuras de datos eficientes.
- `regex`: Manejo avanzado de expresiones regulares para validar y limpiar entradas.
Estas dependencias garantizan una base sólida, ligera y eficiente para desplegar el sistema en entornos con recursos limitados.

12. Instrucciones para ejecutar el bot:

Suponiendo que se cuente con un acceso a internet o se cuente con un servidor en la nube para este propósito, inicialmente se debe crear un bot con Botfather en la plataforma Telegram. Hecho esto, se debe tomar el token brindado por Botfather y pegarlo en el archivo token.txt. Luego, se debe ejecutar startup.bat o startup.sh en dependencia de si se cuenta con un sistema operativo Windows o Linux. Como alternativa para ejecutarlo en un servidor, se puede tomar el comando de startup.sh y vincular el token del bot como argumento. Tras la ejecución de estos pasos se abrirá una consola que representa la ejecución y funcionamiento del bot. Cerrar esta consola interrumpirá el uso del bot en la plataforma. Se recomienda usar servidores dedicados para este propósito. Al iniciar el bot, en caso en que no se muestre un menú con los comandos, se deben insertar manualmente en la configuración de Botfather. A pesar de esto, los comandos funcionarán aunque no sean mostrados por un menú. Esto es una cuestión puramente estética.

13. Pruebas realizadas

Para garantizar la calidad y robustez del bot de recomendación de optativas, se diseñó y ejecutó una batería exhaustiva de pruebas funcionales, estructuradas según los distintos perfiles de usuario (estudiante, profesor y superadmin) y las funcionalidades clave del sistema.
Las pruebas se dividieron en los siguientes bloques:
- Pruebas de acceso y autenticación: se verificaron los flujos de inicio de sesión para profesores y superadmin, la validación de credenciales, la persistencia de sesión por ID de Telegram y el cierre de sesión con restauración del menú inicial.
- Gestión de archivos: se probaron todos los casos posibles de carga de archivos .json, incluyendo validaciones de nombre, estructura, codificación UTF-8 y contenido. Se verificó que los archivos válidos fueran correctamente reemplazados, y que cualquier acción quedara registrada en el archivo de log del sistema.
- Manejo de estudiantes: se probaron los flujos para agregar, visualizar y eliminar estudiantes, tanto de forma individual como masiva (usando la palabra clave 'TODO'). Se incluyó detección de duplicados, manejo de entradas mal formateadas y agrupamiento por grupo.
- Manejo de optativas: se evaluó la creación guiada de optativas mediante flujo conversacional, con validación de nombres duplicados, opción de cancelación y campos obligatorios. También se probaron la eliminación (por nombre y global), la desasignación automática de estudiantes y la visualización formateada.
- Gestión de profesores: el superadmin pudo agregar y eliminar profesores, mientras que los usuarios no autorizados recibieron los mensajes de advertencia correspondientes. También se probó la visualización general de profesores registrados.
- Sistema de reseñas: se evaluó el flujo completo para dejar una reseña, incluyendo validación del estudiante, optativa asignada, puntuación válida (1-5), reemplazo de reseñas existentes y almacenamiento

en disco. Además, se probó la consulta de reseñas por optativa y la eliminación total de reseñas por el superadmin.
- Motor de búsqueda: se probaron búsquedas por texto libre con soporte para ponderación (***) y exclusión (!palabra). El sistema generó resultados basados en similitud semántica mediante un modelo TF-IDF, devolviendo información detallada por optativa, incluyendo descripción, plazas disponibles, asignaturas relacionadas y las mejores y peores reseñas, todo en mensajes separados para evitar límites de Telegram.
- Comandos y flujo conversacional: se testearon todos los comandos disponibles (/start, /help, /login, /rev, /vrev, /log, /delrev), así como sus comportamientos según el rol del usuario. El menú dinámico del profesor también fue verificado en su totalidad.
- Validación y tolerancia a errores: se incluyeron pruebas deliberadas con datos malformados, credenciales erróneas, comandos fuera de lugar y cancelaciones para comprobar la resiliencia del bot y la claridad de sus mensajes ante fallos.
- Registro y trazabilidad: se confirmó que toda acción administrativa (agregados, eliminaciones, modificaciones y cargas de archivo) quedara documentada correctamente en el archivo registro_operaciones.txt, manteniéndose dentro del límite de 1000 entradas más recientes.

Estas pruebas se ejecutaron manualmente desde múltiples cuentas de Telegram. El sistema demostró ser estable, funcional y seguro, cumpliendo con todos los objetivos definidos en la etapa de diseño.

14. Anotaciones Finales

El desarrollo de este sistema demostró ser una experiencia integral, donde se combinaron múltiples aspectos técnicos del desarrollo de software moderno, incluyendo manejo de estructuras de datos, diseño conversacional, autenticación, persistencia de información, interacción con archivos, y un motor de búsqueda personalizado basado en técnicas de recuperación de información.
Uno de los mayores retos fue lograr una interfaz conversacional intuitiva y robusta que atendiera tanto a estudiantes como a profesores, con flujos diferenciados según sus permisos. Se prestó especial atención al manejo de errores, a la validación rigurosa de los archivos y a la trazabilidad de las operaciones administrativas, asegurando así la integridad y confiabilidad del sistema.
También fue clave implementar un motor de recomendación semántico que fuera más allá de simples coincidencias de texto, permitiendo búsquedas ponderadas e interpretables. El sistema de reseñas aportó un valor añadido, fomentando la participación estudiantil y enriqueciendo la calidad de las recomendaciones.
Este proyecto no solo cumplió con los requisitos funcionales establecidos, sino que dejó sentadas las bases para futuras extensiones, como el uso de agentes inteligentes, integración con bases de datos externas, ampliación a otras plataformas o incluso incorporación de modelos de lenguaje avanzados para respuesta generativa.
En resumen, se entrega un sistema funcional, validado y completo, que representa una solución práctica y útil para la gestión y recomendación de optativas en entornos educativos.