Conclusiones

El desarrollo del proyecto de biblioteca virtual permitió aplicar múltiples conceptos fundamentales del desarrollo de software, tanto el backend como el frontend, así como el diseño de la base de datos y la gestión de la lógica de negocio.

La elección de una base de datos relacional SQLite, junto con el uso de Peewee como ORM, facilitó la correcta modelación de entidades como libros, autores, géneros, editoriales, usuarios y préstamos, respetando principios de normalización y evitando redundancias innecesarias. El uso de tablas intermedias para relaciones muchos a muchos(N-M) permitió una estructura flexible y extensible, especialmente en casos como libros con múltiples autores o géneros. Ademas, se evaluaron criterios como el uso de claves primarias, claves foráneas, restricciones de unicidad e índices, concluyendo que, aunque el volumen de datos no sea masivo, ciertos índices (por ejemplo en campos de búsqueda frecuente como títulos o correos electrónicos) pueden aportar beneficios sin agregar complejidad excesiva.

El uso de Jinja2 para la construcción de los templates HTML fue un punto clave en el proyecto, ya que permitió aplicar lógica de presentación de manera clara y ordenada, manteniendo una separación adecuada entre la vista y la lógica de negocio. Gracias a su sintaxis inspirada en Python, fue posible trabajar con condiciones, bucles y variables de forma intuitiva, facilitando la renderización dinámica de datos como estados de libros, listados por género, mensajes flash y componentes reutilizables. Esto contribuyó a una interfaz más flexible, legible y fácil de mantener, incluso a medida que la complejidad del sistema aumentaba.

En cuanto a la lógica del sistema, se optó por un enfoque donde muchos estados del libro se determinan dinámicamente en función de su situación real (disponible, reservado, en uso, atrasado y no disponible), lo cual reduce la dependencia de estados rígidos y minimiza inconsistencias. Este enfoque mejora la confiabilidad de la información y simplifica las consultas, a costa de una lógica ligeramente más compleja pero legible.

El uso de Flask-Admin permitió acelerar el desarrollo de interfaces administrativas, concentrando los esfuerzos en la lógica del dominio en lugar de en la construcción manual de formularios complejos. Al mismo tiempo, se aprendió a personalizar su comportamiento para casos no triviales, como cargas de archivos, validaciones avanzadas, relaciones complejas y presentación visual personalizada, lo que refuerza la comprensión del framework más allá de su uso básico.

En materia de seguridad y autenticación, se abordaron prácticas adecuadas como el hashing de contraseñas, la verificación por correo electrónico, la recuperación de contraseñas mediante tokens temporales y la prevención de abuso mediante límites y validaciones. Estas decisiones reflejan una preocupación por la seguridad real del sistema,

incluso en un contexto académico o de práctica, lo cual resulta fundamental para una formación sólida.

Respecto al uso de inteligencia artificial, se empleó IA supervisada como herramienta de apoyo puntual**,** principalmente para la resolución de dudas técnicas específicas, validación de enfoques y generación de fragmentos pequeños de código o lógica. No se delegó la construcción completa de funcionalidades críticas a la IA, sino que se utilizó como un recurso complementario, manteniendo el control del diseño, la arquitectura y las decisiones finales. Este enfoque permitió mejorar la productividad y el aprendizaje sin comprometer la comprensión del sistema ni la autoría del código.

Durante el desarrollo de este proyecto se evidenció la importancia de adoptar un enfoque progresivo y realista al momento de comenzar a trabajar. Iniciar el proceso de manera lenta, incluso con tareas mínimas como abrir los programas necesarios, revisar la estructura general o definir objetivos simples, permitió reducir la fricción inicial y evitar la sensación de bloqueo que suele aparecer frente a proyectos más amplios. La división del trabajo en objetivos pequeños y alcanzables facilitó un avance constante, donde cada paso cumplido generó mayor claridad sobre el siguiente, haciendo que el desarrollo ganara ritmo de forma natural con el tiempo.

Este método de comenzar por lo más sencillo resultó clave para construir confianza sobre el sistema, ya que permitió validar decisiones tempranas sin necesidad de resolver toda la complejidad desde el inicio. A medida que las bases estuvieron firmes, fue posible incorporar funcionalidades más complejas con mayor seguridad y comprensión del código existente. De esta forma, el proyecto no solo avanzó de manera ordenada, sino que también promovió una mejor organización mental del problema, demostrando que un inicio pausado y bien estructurado puede ser más efectivo que intentar abarcar todo desde el primer momento.

Finalmente, el proyecto demuestra que incluso aplicaciones de tamaño moderado pueden beneficiarse de buenas prácticas como separación de responsabilidades, claridad en los modelos, decisiones conscientes sobre almacenamiento y estados, y una arquitectura pensada a futuro. Más allá del resultado funcional, el mayor valor del proyecto reside en el proceso de análisis, decisión y refinamiento continuo, sentando una base sólida para proyectos de mayor escala y complejidad.