ENTREGA AB FINAL

Alexandr.IA

28 DE MAYO DE 2024

PABLO NICOLÁS SOTO IRAGO
Computer Science & AI

Índice

Alexandr.IA

Descripción

La aplicación Alexandr.IA es una aplicación especializada para la gestión de libros del usuario. Se busca a un público objetivo, muy amplio, que abarca a todas las edades (que puedan manejar un dispositivo móvil), los cuales serán lectores o estudiantes que quieran o necesiten organizar y clasificar sus libros de forma intuitiva. El propósito de la aplicación es que los usuarios puedan almacenar datos de sus libros, además de donde se encuentran y porque página van, en una aplicación que no necesite muchos recursos. Alexandr.IA facilita una forma de organizar sus libros cómoda y visualmente.

Investigación de software

Una investigación de software es un proceso fundamental que debemos hacer antes de iniciar un proyecto de software. En esta investigación se analizarán y evaluarán las tecnologías, herramientas y frameworks para cumplir de la mejor forma todas las necesidades del proyecto. Para realizar una buena investigación de software, es necesario seguir una serie de pasos:

Definición de requisitos

Para realizar correctamente la investigación, debemos entender al completo cuáles van a ser las acciones que podrá realizar Alexandr.IA. Por ello, debemos definir una serie de requisitos:

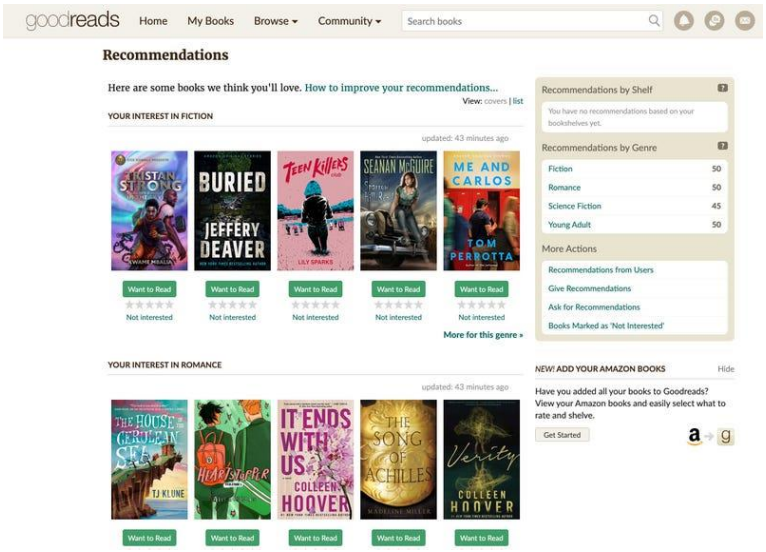
El software debe ser capaz de crear, leer, actualizar y eliminar libros (CRUD). Permitirá apuntar el título del libro, su categoría, una descripción del libro, la ubicación donde se encuentra y la página. Todos los libros deben tener la posibilidad de asignarles un icono para diferenciarlos de forma más gráfica. Además, se añadirá una función que posibilite compartir libros con otros usuarios dentro de la aplicación. Se quiere añadir una función de reconocimiento por voz, que permitirá al usuario añadir los libros sin tener que abrir la aplicación. La función de la inteligencia artificial se añadirá como herramienta que sugerirá libros al usuario según los libros que haya añadido a su lista.

Se añadirán funciones de compraventa de libros, dónde exista una página web donde comprar libros de librerías externas, y una función en la aplicación de compraventa entre usuarios, con posibilidad de publicar reseñas y puntuaciones.

Análisis de mercado

Debemos realizar un estudio de mercado para valorar si hay posibilidades de introducirse en el mercado de las aplicaciones de gestión de bibliotecas.

Proponemos una idea existente en el mercado, pero queremos centrarnos mucho más en la gestión de bibliotecas a baja escala, ya que se busca gestionar simplemente los libros que pueda tener un usuario en posesión. La complejidad de destacar en el mercado reside en la alta influencia que tienen actualmente otras aplicaciones parecidas como GoodReads o Handy Library – Book Organizer:



Aun así, valoramos que el mercado necesita una herramienta fácil de manejar, que permita al consumidor organizar sus libros y aumentar así su tiempo de lectura. Nuestra idea principal se relaciona con la oportunidad de mercado que tenemos, ya que casi cualquier persona tiene libros en casa, no importa la categoría, que necesitan organizarse de alguna forma. Nuestra diferenciación se basa en la mezcla de estas 2 aplicaciones, donde se juntan las mejores funcionalidades como la organización digital y el catálogo de nuevos libros. Además, con las funciones de compraventa y de sugerencias con IA, tenemos cierta ventaja respecto a la competencia.

Claramente, nuestro público objetivo serán lectores de todas las edades, estudiantes y lectores ávidos, además de personas con conocimientos básicos sobre las nuevas tecnologías.

Evaluación técnica

Debemos evaluar los aspectos técnicos, las limitaciones y las capacidades de la aplicación.

1. Seguridad y Privacidad:

En las futuras actualizaciones de Alexandr.IA se implementará la tecnología blockchain para garantizar la seguridad de los datos de cada usuario, siendo solamente accesibles por el administrador, el usuario y a quién se le permita el acceso. Además, se utilizará un cifrado automático para proteger la información sensible.

2. Interoperabilidad:

La aplicación facilitará poder compartir libros entre usuarios y próximamente a grupos. Se incluirá una sincronización con otras plataformas de compra de libros para poder generar catálogos reales.

Tecnologías:

Se utilizará el framework Flutter con lenguaje dart para el desarrollo de una aplicación multiplataforma. En la siguiente tabla se definirán algunos pros y contras de Flutter:

VENTAJAS	DESVENTAJAS
Buen rendimiento gracias a su motor gráfico, que permite desarrollar aplicaciones de forma rápida y bien optimizado.	Tamaño de las aplicaciones de desarrollo, no es el caso de la versión beta de Alexandr.IA.
El tiempo de desarrollo es menor comparado con la competencia ya que proporciona herramientas únicas que reducen el tiempo de desarrollo.	La popularidad de dart es baja, comparada con Java, JavaScript o C#, aunque poco a poco está ganando popularidad.
Flutter es un framework muy fácil de aprender, además de ser muy visual para programadores sin experiencia.	Posibles problemas de compatibilidad con iOS, aunque sea un lenguaje multiplataforma.

Viabilidad económica

Ya hemos visto que la idea de Alexandr.IA frente a la competencia es mucho mejor, por lo que debemos seguir desarrollando el proyecto. Para realizarlo, no vamos a necesitar unos gastos económicos altos, ya que el desarrollo de la aplicación no es muy complejo, pero según los resultados en el mercado de la aplicación, se podrían implementar sistemas de mantenimiento y soporte técnico. En el caso de necesitar servicios adicionales, se ha pensado el próximo plan de viabilidad económica.

1. Desarrollo de Software: Para continuar actualizando la aplicación y la interfaz de la aplicación, se requerirá de un equipo de mantenimiento, el cuál dependería de los ingresos que genere la aplicación. El precio del mantenimiento podría ser de 3.200€ al mes (según el equipo contratado). Al año podría costar alrededor de 42.000€.

2. Infraestructura y Tecnología: La aplicación no requiere de hardware complejo para su correcto funcionamiento, pero según la cantidad de usuarios, se podrían implementar PaaS, que ofrecería una plataforma en la nube para desarrollar y gestionar la aplicación. En este caso, se utilizará Kasm Workspaces, que ofrece una versión gratuita, que incluye tanto la seguridad de datos como una API. Si además queremos implementar servicios externos de compra e IA, los costes totales anuales podrían ascender a 25.000€. Implementar una IA podría costar unos 12.000€, principalmente por los elevados costes de los datos. Los costes de la función del catálogo dependerán de las tarifas que tengan las empresas de libros. Después de una larga investigación, se establece que las empresas cobran unas tarifas mensuales por acceder a sus bases de datos. los costes totales en un año podrían ascender a 13.000€.

3. Cumplimiento Normativo en Múltiples Jurisdicciones: Cumplir con la legislación, en este caso europea, de privacidad y seguridad de datos no resultaría un problema, ya que la aplicación estará actualizada frente a posibles cambios de la ley de protección de datos del consumidor, además de muchas otras relacionadas con la privacidad de datos del usuario. No reflejará ningún coste adicional, ya que se encargaría el equipo de mantenimiento contratado.

En total, la inversión inicial del primer año tendría un coste de 67.000€. Estimamos estos gastos en un año pensando en que la aplicación tenga un ROI (retorno de inversión) positivo del 30%, y basándonos en los datos de descargas de HandyLibrary, se podría recuperar esta inversión con 300.000 descargas entre Android y iOS.

Compatibilidad e integración

Como aplicación nueva en el mercado, no planteamos integraciones a corto plazo. Si que es cierto que, al querer implementar herramientas de compraventa, deberíamos implementar servicios de Google que puedan dar acceso a las tiendas de libros.

El principal aspecto de compatibilidad que se debe valorar es el correo electrónico. El correo será la única forma de guardar tus datos, y utilizar la aplicación en otros dispositivos. Así se implementarán protocolos de seguridad adicionales que permitirán al usuario mantener los mismos datos entre dispositivos con la misma cuenta de usuario.

La compatibilidad entre dispositivos no resultará un problema para la aplicación. Alexandr.IA está desarrollada con Flutter (Dart), por lo que ya posee un software compatible con Android y iOS.

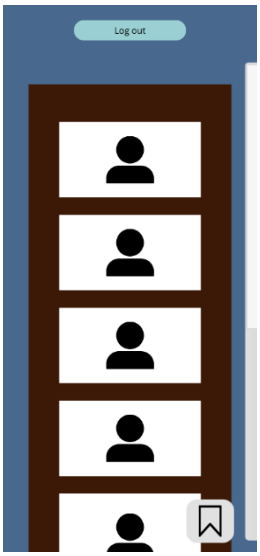
Pruebas y prototipos

PANTALLA INICIO /LOG IN

PRIMERA PANTALLA

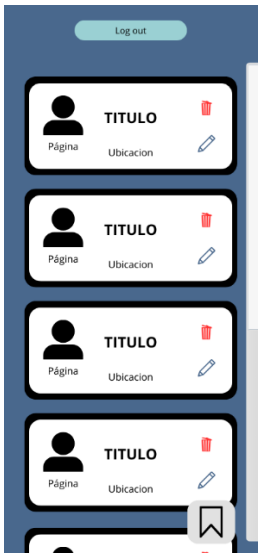
SEGUNDA PANTALLA

Organización de los libros por icono



VER LIBROS

R y D del CRUD



PANTALLA EDITAR

U del CRUD

PANTALLA CREAR

C del CRUD

Estudio de viabilidad, marco TELOS

El propósito de realizar un estudio de viabilidad del proyecto es para evaluar la factibilidad del proyecto, la planificarlo estratégicamente y reducir los futuros riesgos. Utilizando el modelo TELOS, se puede analizar más a fondo la viabilidad y a garantizar una implementación exitosa. Para ello, profundizaremos en los desafíos tecnológicos, económicos, legales y operativos que tiene este proyecto para valorar todos los campos que afectan al proyecto.

Tecnología

La tecnología requerida por el proyecto no es muy compleja y actualmente disponemos de todas las herramientas tecnológicas necesarias para llevar a cabo este proyecto. Se necesita usar el framework Flutter, una herramienta gratuita que utilizaremos a través de Visual Studio Code. En un principio, utilizaremos servicios gratuitos de bases de datos como SQLite.

Como principal desarrollador del proyecto, poseo los conocimientos básicos para programar la aplicación en Flutter, pero necesitaría un equipo adicional para trabajar con una base de datos.

Economía

La aplicación inicial no necesitará ningún gasto en desarrollo de software, pero en un futuro provisional que dé buenos resultados, se podrían contratar servicios adicionales como un equipo de mantenimiento, servicios de IA y compra, y servidores en la nube. En el caso de obtener ingresos, estimamos que los gastos en mantenimiento serán de un 70% de esos ingresos, siendo los ingresos mayores o iguales a 100.000€. Pensando en el ROI, si un 70% se destinan al mantenimiento y actualizaciones de la aplicación (gastos), nos quedarían alrededor del 30% de beneficios.

Legal

En España y Europa están en vigor dos principales leyes sobre la protección de datos, las cuales son LOPD en España, y RGPD en Europa. Este cumplimiento de la ley en la aplicación será gestionado por el equipo de mantenimiento. Los contratos y acuerdos con terceros deberán ser examinados para que no se permita el incumplimiento de la ley actual.

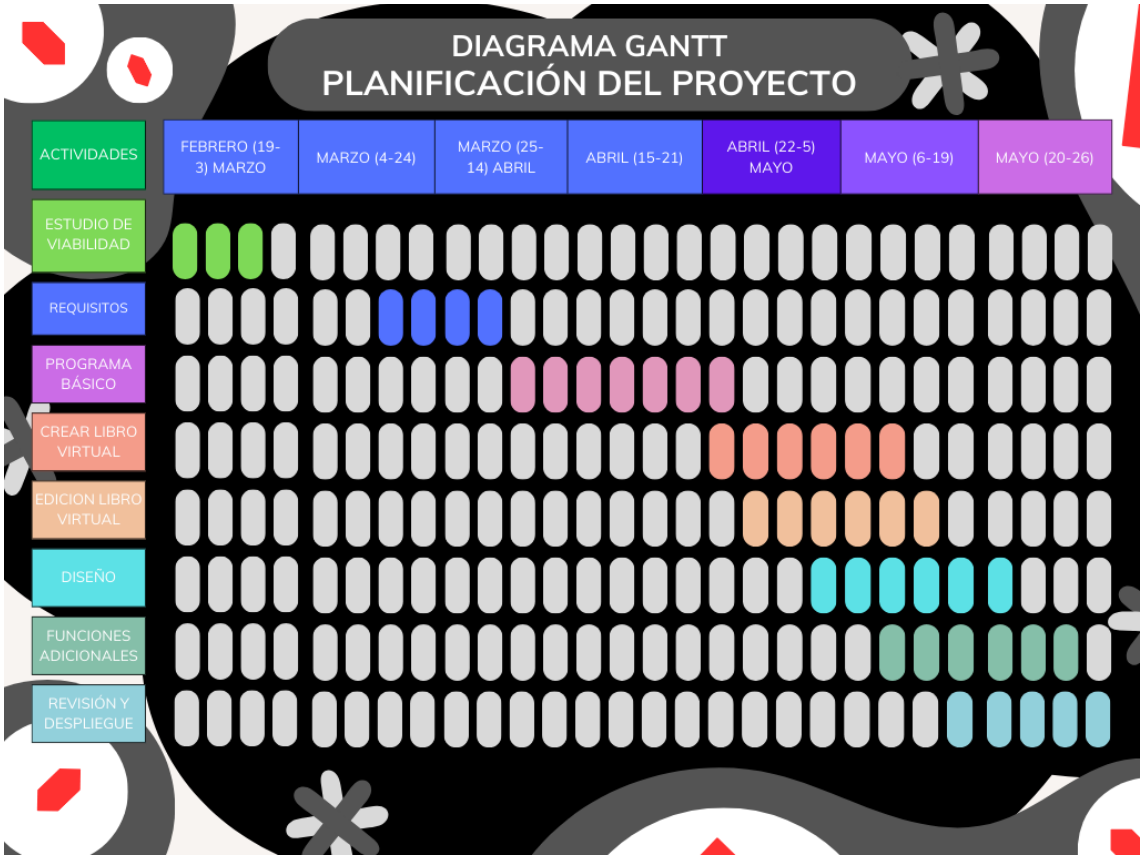
Por ahora, a corto plazo, no planteamos acuerdos con terceros sobre nuestros servicios, este aspecto se valorará en un futuro donde la aplicación tenga relevancia en el mercado.

Operacional

Actualmente, los desafíos operacionales no resultan ningún problema ya que tenemos la capacidad para mantener el proyecto. La capacidad operativa del equipo de mantenimiento y desarrollo es esencial para el éxito del proyecto. Según su futuro, se deberá evaluar una adaptación de personal y sistemas a los nuevos requisitos que surjan. Entendemos que las estrategias y decisiones que tomemos para abrirnos al mercado deben de servirnos para prevenir problemas, organizar y clasificar nuestros objetivos e intentar no improvisar soluciones de problemas importantes.

Planificación

El desarrollo de la aplicación se dividirá en varias etapas:



En este cronograma se tiene en cuanta posibles riesgos que paralicen o retrasen la planificación del proyecto. Además, se debe revisar periódicamente para reajustar las fases del desarrollo y así mantener un cronograma óptimo.

También es importante establecer la metodología que vamos a usar. Esta dependerá de factores como la flexibilidad del proyecto, las tecnologías o herramientas, la experiencia del equipo, etc. Por ello vamos a valorar dos modelos iterativos y dos secuenciales de ciclo de vida del desarrollo de software.

- En cuanto a los modelos iterativos, podemos hablar sobre las metodologías ágiles o "agile", que se guían por las necesidades del cliente, para cumplir sus expectativas. Siempre se busca trabajar eficientemente con esta metodología a través de eliminar tareas innecesarias y de una mejora y reflexión continua. Dentro de esta metodología podemos identificar otras como Scrum o Kanban.
- Los modelos secuenciales se caracterizan por establecer unas fases desde el principio, con una planificación completa y una documentación exhaustiva. Dos metodologías muy usadas que tienen estas características son Cascada y Espiral.

Para Alexandr.IA, según nuestros requerimientos, necesidades y posibilidades, nos interesa usar la metodología de cascada, ya que se basa en completar una serie de fases ordenadas dentro de un plazo estimado, donde se ha planeado todo desde el principio.

Cascada Etapa 1: Requisitos y documentación

1. Introducción

1.1 Propósito del documento

En este documento se especifican detalladamente los requisitos funcionales y no funcionales de Alexandr.IA, las interfaces del sistema, las restricciones, etc. de la aplicación. Se especificarán todos los requisitos en diferentes apartados.

1.2 Alcance del producto o sistema

Esta aplicación funciona tanto en Android como en iOS. La aplicación de gestión de libros denominada "Alexandr.IA" proporciona al usuario una herramienta para gestionar su colección de libros actuales, y así ayudar al usuario a localizarlos. Se podrá crear un apartado donde añadir los libros y clasificarlos, se podrán modificar y eliminar siempre que el usuario quiera. Además, tendrá un apartado de compra donde se podrá acceder a una tienda online de libros. Esta tienda online será dependiente de las librerías y bibliotecas de la zona donde reside el usuario. En las versiones más avanzadas de la aplicación se podrá usar una inteligencia artificial para ofrecer al usuario opciones sobre que libro leer, según las categorías más leídas. La aplicación ofrecerá la conexión con un correo electrónico, para sincronizar los datos de la aplicación con otros dispositivos y guardar los datos en la nube. También se implementarán funciones de bibliotecas compartidas, conectándose remotamente mediante la cuenta del usuario.

1.3 Definiciones, acrónimos y abreviaturas

Libro virtual (2.2): Libro que crea en su dispositivo cada usuario con sus correspondientes características.

Bibliotecas compartidas (1.2): Lista de libros virtuales compartidos entre 2 o más usuarios de la plataforma.

UI (2.2): Interfaz de Usuario.

Bibliotecas virtuales (2.3): Lista de libros virtuales que comparten alguna característica, normalmente la ubicación.

1.4 Referencias a otros documentos relevantes

Documentación de Flutter

Diagrama de casos de uso:

https://creately.com/blog/es/diagramas/tutorial-diagrama-caso-de-uso/

Diagrama de flujo:

https://miro.com/es/diagrama-de-flujo/que-es-diagrama-de-flujo/

Diagrama de estado:

https://www.edrawsoft.com/es/uml/uml-state-diagram.html

1.5 Visión general del documento

La aplicación Alexandr.IA ofrecerá al usuario una fácil e intuitiva gestión de sus librerías personales. Esta aplicación se desarrolla con el propósito de proporcionar al usuario una herramienta con la que se pueda organizar y gestionar los libros de forma interactiva y/o remota, permitiendo un acceso rápido a los libros guardados, además de sus funciones integradas. Los principales beneficios de está aplicación se centran en la comodidad del usuario, aumentando la productividad de este (mediante la organización de los libros), aumentando la flexibilidad del usuario (pudiendo gestionar los libros a distancia) y reduciendo el estrés, dando al usuario cierta paz mental en el momento que busque una nueva lectura.

2. Descripción general

2.1. Perspectiva del producto

La aplicación tendrá las opciones de ser tanto dependiente como independiente do otros sistemas. En la parte dependiente, la gestión de los libros en posesión del usuario no necesitará conexiones externas. En cambio, la parte independiente se refiere a que la gestión de los libros podrá incluir funciones como la ubicación (tanto del dispositivo, como la ubicación que se le establezca al libro); la conexión a internet, para vender libros en posesión o comprar libros en tiendas online. En la función de bibliotecas compartidas, se necesitará una conexión online para contactar con los participantes de la biblioteca compartida.

2.2. Funciones del producto

- Las principales funciones de la aplicación serían que permita crear un “libro virtual”, actualizarlo, poder leerlo y borrarlo cuando decida el usuario. Permitirá apuntar el título del libro, su categoría, una descripción del libro, la ubicación donde se encuentra y la página.

- La aplicación debe tener acceso a la red para usar las funciones de compraventa y bibliotecas compartidas.

- Se podrán también crear ubicaciones virtuales donde situar los libros, y así tener gráficamente donde se encuentra el libro. El usuario también podrá establecer la ubicación del libro mediante satélite.

- Las funciones de estilo permitirán establecer una interfaz adaptada al usuario, permitiendo cambiar el tamaño de los objetos, de la letra y de la pantalla; los colores de cada elemento, y los estilos de la UI (modo claro y oscuro).

2.3. Características del usuario

Los usuarios serán principalmente serán lectores y estudiantes, además de personas con conocimientos básicos sobre las nuevas tecnologías, donde se busca un público con grandes colecciones de libros. La aplicación ofrecerá todos los idiomas que se usan actualmente en el mundo (con posibilidad de solicitar la integración de nuevos idiomas o dialectos).

2.4. Restricciones generales

Alexandr.IA será completamente compatible con Android e iOS. En cuanto a los recursos tecnológicos disponibles, como la memoria el almacenamiento, o el poder de

procesamiento serán muy bajos, aunque, según los datos que se vayan creando y guardando en la aplicación podrían ir necesitando más recursos. Se tendrán en cuenta las leyes que puedan afectar al desarrollo del proyecto, sobre todo las leyes de protección de datos. Estas leyes serán cumplidas a través de una encriptación de los datos, cumpliendo los requisitos propuestos de seguridad, además de integrar una protección contra las posibles vulnerabilidades.

2.5. Supuestos y dependencias

Hay factores que damos por supuestos como el cumplimiento de los requisitos de hardware y software, que estarán relacionados con la versión del dispositivo y la posibilidad de instalar la aplicación. Además, se da por sentado que existe una conexión a internet para descargarla y poder realizar ciertas acciones avanzadas en la aplicación. Dentro de las dependencias, Alexandr.IA dependerá de algunos servicios en la nube como SQLite (para almacenar los datos del usuario), dependerá de actualizaciones periódicas, y dependerá de la opinión del usuario para implementar futuros cambios y mejoras.

3. Requisitos

Los requisitos de software es lo que define el funcionamiento y el objetivo (además de el alcance) de un programa de software, en este caso de Alexandr.IA. Se explicarán los requisitos de este programa según su funcionalidad y su comportamiento:

3.1. Requisitos funcionales

- RF01: Acceder a la aplicación en modo sin conexión, y mantener los datos elegidos siempre disponibles.
- RF02: Crear un “libro virtual”, actualizarlo, poder leerlo y borrarlo cuando decida el usuario.
- RF03: Tener acceso a la red para usar las funciones de compraventa y bibliotecas compartidas.
- RF04: Permitir al usuario establecer la ubicación satelital de sus libros. Se podrá también crear una ubicación virtual, para situar sus libros, y así tener gráficamente donde se encuentra cada libro.
- RF05: Permitir al usuario modificar y adaptar la UI, permitiendo cambiar el tamaño de los objetos, de la letra y de la pantalla; los colores de cada elemento, y los estilos de la interfaz (modo claro y oscuro).

3.2. Requisitos no funcionales

- RNF01: La aplicación debe de tener un buen rendimiento para cargar las funcionalidades en poco tiempo (menor a 3 segundos).
- RNF02: Los datos del usuario siempre estarán encriptados, además de incluir medidas de autenticación segura como contraseñas fuertes (requeridas por el sistema) un sistema de autenticación en 2 pasos.
- RNF03: Desde un principio, la aplicación tendrá una interfaz intuitiva que podrá ser modificada externamente por el usuario (cambios de estilo).
- RNF04: La aplicación será compatible con cualquier dispositivo Android o iOS.
- RNF05: El sistema estará preparado para soportar el aumento significativo de usuarios y mantendrá un rendimiento acorde a los requerimientos.

3.3. Requisitos de datos

- RD01: La aplicación recogerá datos tanto en el sistema como en la nube (según decida el usuario). Todos ellos bajo cifrado. Los datos con estas características son:

  - Los datos del usuario (únicamente los introducidos por el mismo), como el nombre de usuario o el correo electrónico.
  - Los libros virtuales, como el título o la ubicación.
  - La configuración de la aplicación (tanto de la UI como la del usuario).
  - Los datos de sesión y sincronización.
  - Los datos de seguridad (claves y formas de recuperar los datos).

- RD02: Los datos se analizarán y validarán para omitir fallos y errores en el sistema.

- RD03: La aplicación siempre tendrá la opción de hacer copias de seguridad para permitir al usuario recuperar los datos de su cuenta.

3.4. Requisitos de Sistema

- RS01: La aplicación es totalmente compatible con Android y iOS en sus versiones más recientes. Mientras la aplicación se desarrolle en Flutter (Dart), no hay problema.

- RS02: Se necesitará un acceso continuo a la red para realizar operaciones dentro de la aplicación.

- RS03: Los servidores deben estar disponibles en todo momento (menos en periodos de mantenimiento), y soportar un aumento de usuarios en el sistema.

3.5. Requisitos de Interfaz

- RI01: La interfaz debe seguir las directrices de diseño de Material Design, para que la UI sea coherente y accesible. Debe tener un diseño responsivo y adaptativo para cada dispositivo.

- RI02: El estilo de la interfaz se debe de poder modificar, para mejorar la experiencia del usuario.

- RI03: La aplicación se integrará con APIs para la actualización y consulta de datos.

3.6. Requisitos de Operación y Mantenimiento

- ROM01: La aplicación debe de ser fácil de instalar e iniciar. Requerirá de unos ajustes básicos del usuario de su cuenta en la aplicación.

- ROM02: Se requiere una documentación para las actualizaciones, parches y mantenimientos de la aplicación.

- ROM03: La aplicación contará con herramientas de monitoreo que permitirán supervisar el rendimiento del sistema y detectar futuros problemas.

-ROM04: Se creará un soporté técnico, según la rentabilidad de la aplicación, para resolver posibles problemas en el sistema o del usuario.

3.7. Requisitos de Usuario

- RU01: El usuario no necesitará conocimientos informáticos, pero si un mínimo de experiencia tecnológica, donde el usuario sepa manejar un dispositivo móvil con facilidad.

- RU02: Los administradores necesitarán una formación que les permita administrar fácilmente la aplicación.

3.8. Requisitos de Entrega

- RE01: La aplicación debe tener la versión beta acabada para la fecha de entrega establecida (entrega del AB).

-RE02: La aplicación se podrá descargar en la misma fecha, aunque sufrirá modificaciones y mejoras en versiones futuras.

3.9. Requisitos de aceptación

Los criterios de aceptación de la aplicación son esenciales. Se requieren pruebas de rendimiento y de usabilidad. Las pruebas de rendimiento evalúan cómo funciona bajo diferentes cargas de trabajo, mientras que las pruebas de usabilidad se centran en la experiencia del usuario. Estas pruebas aseguran que la aplicación sea eficiente y fácil de usar.

4. Modelos del Sistema

4.1. Modelos de datos

La base de datos se desarrollará en MySQL, que incluirá tablas como datos_usuario, libros_virtuales,bibliotecas_virtuales y categorías.+

Nos interesa crear un software basado en datos, ya que puede mejorar la fiabilidad y efectividad del programa de diversas maneras, gracias a su capacidad de utilizar grandes volúmenes de información para optimizar procesos, prevenir problemas y mejorar la toma de decisiones. El software basado en datos mejora la fiabilidad y efectividad a través de:

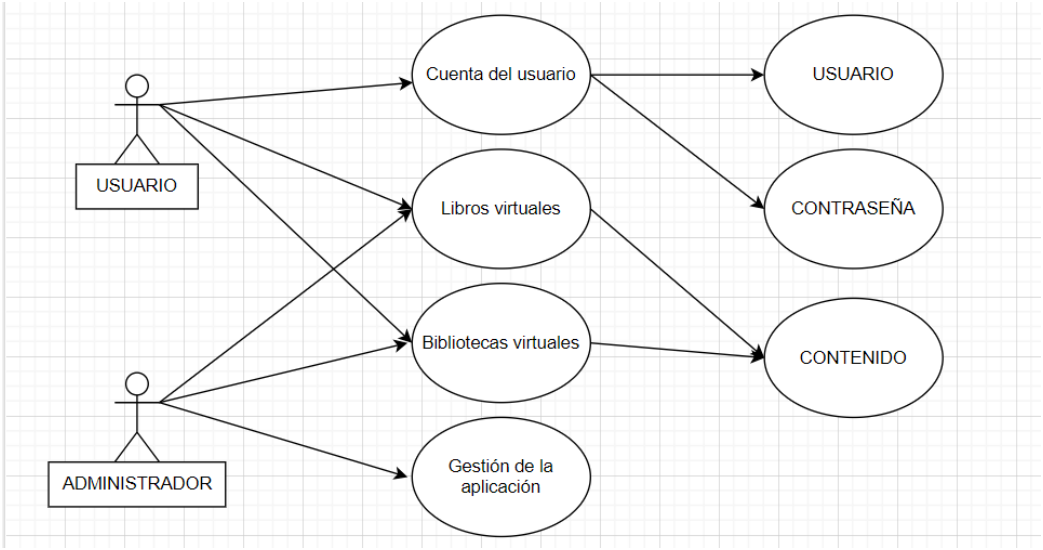
- La rapidez en la recopilación de datos.
- Permitirnos analizar la situación futura y así prever futuros problemas.
- Facilitar la comprensión de la situación a través de patrones o tendencias comparativas a largo plazo.
- Darnos información sobre el usuario para poder personalizar y mejorar su experiencia de uso.

4.2. Máquinas de estado

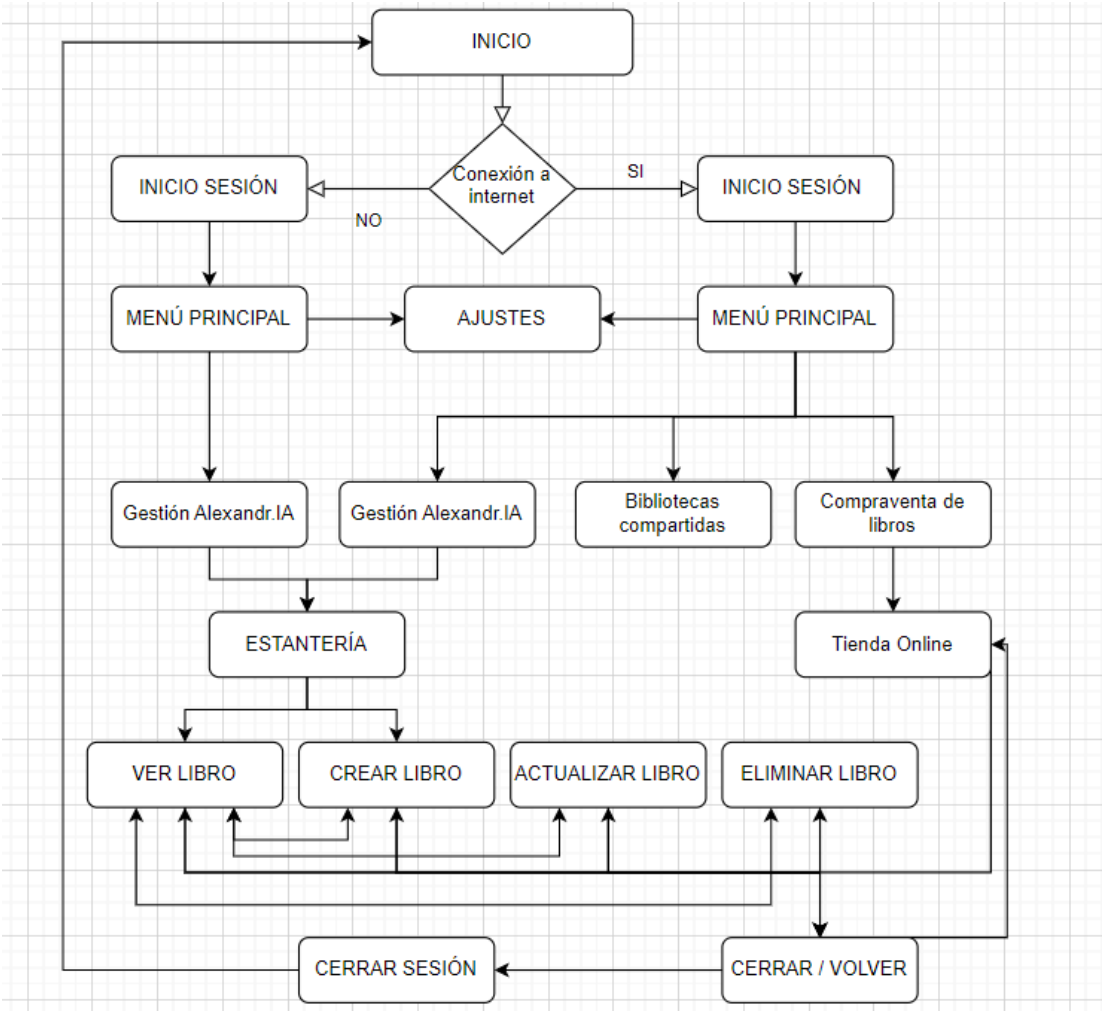
Una máquina de estado es un modelo teórico usado en el desarrollo de software para representar el comportamiento del sistema a través de estados y transiciones. Se aplica en el diseño, implementación de controladores y validación de software para asegurar su correcto funcionamiento según los requisitos. Se usan principalmente para simular el comportamiento de sistemas en diversos estados y momentos, que pueden cambiar de estado según eventos. También, las máquinas permiten gestionar el estado de un sistema ordenadamente, facilitando la implementación, el diseño y el mantenimiento.

Podemos encontrar las máquinas de estado finitas (FSM), las máquinas de estado infinitas y las máquinas de estado finita extendida (EFSM). Los nombres se asocian con el conjunto de estados que puede manejar cada máquina. La diferencia entre la FSM y la EFSM es que en la extendida tiene estados finitos, que pueden cambiar a otros estados directamente sin tener que pasar por los demás, mientras que en la FSM tiene unas transiciones predefinidas entre los estados.

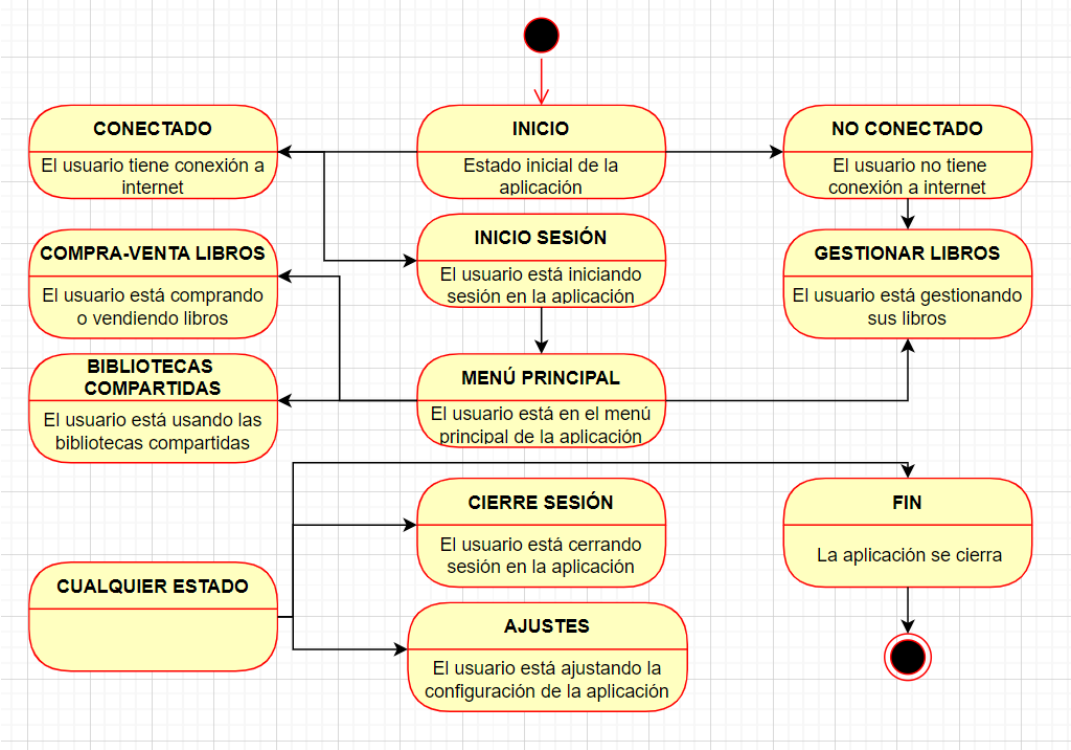
4.3. Diagramas de Casos de Uso



4.4. Diagramas de flujo



4.5. Diagramas de estado



5. Verificación y Validación

5.1. Plan de V&V

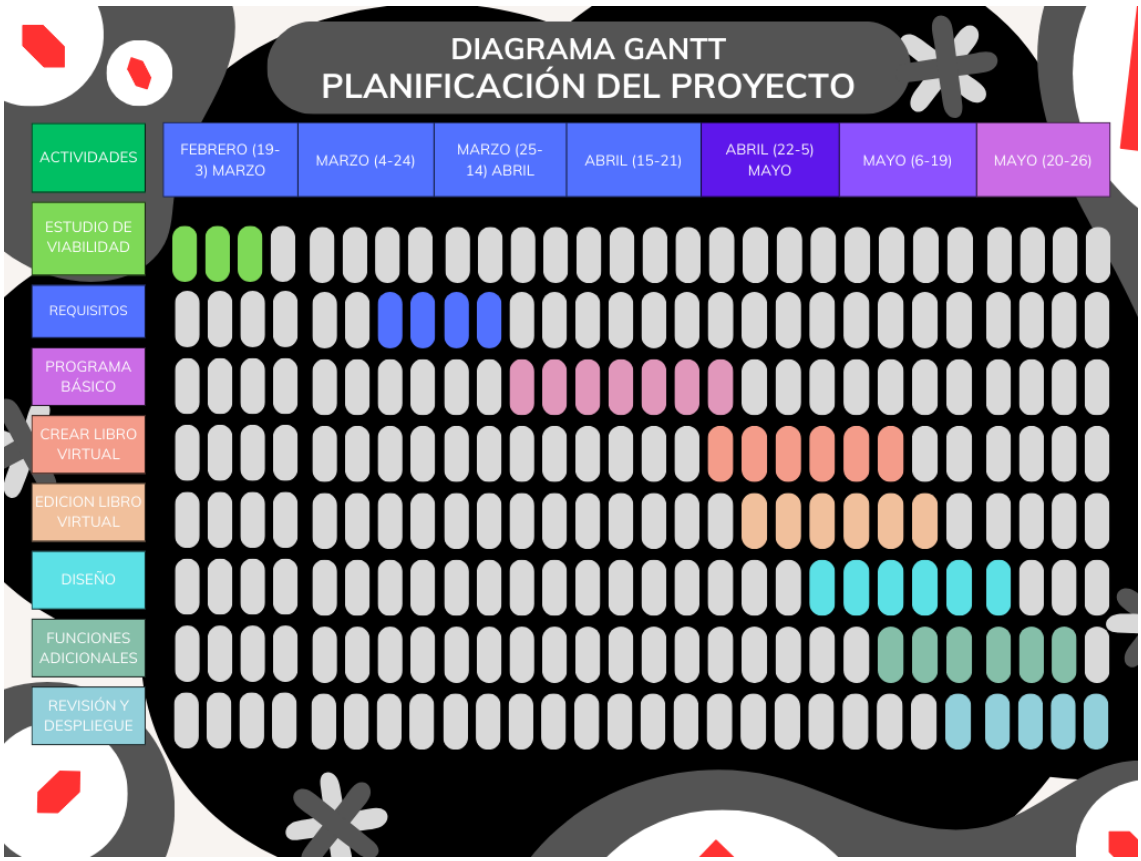
La implementación de procesos para verificar y validar el software es necesario para garantizar su calidad y funcionalidad. Las pruebas unitarias se centran en evaluar componentes individuales del software, mientras que las pruebas de integración examinan cómo interactúan estos componentes juntos. Por último, las pruebas de usuario permiten validar la experiencia del usuario final. Estos procesos combinados aseguran que el software sea robusto y cumpla con los requisitos establecidos.

6. Apéndices

6.1. Información de Soporte

Material adicional como FAQ, guías de usuario y protocolos de reporte de errores.

Diagrama de Gantt



Diseño

Traducir requisitos en soluciones técnicas

Interpretación de Requisitos

En los requisitos que se implementarán seguro dentro del plazo estimado, la aplicación podrá crear, leer, modificar y eliminar los libros virtuales, incluyendo características como el título, la descripción, la ubicación, la categoría, etc.

Diseño de código

Se organizará por carpetas, donde se separarán las pantallas secundarias, las imágenes, los datos y todo lo necesario para crear la aplicación. Para ello se usará una programación orientada a objetos, definiendo clases, atributos y métodos. Las principales clases estarán relacionadas con el CRUD, cuyos métodos se podrán utilizar en diferentes archivos “.dart”. Los requerimientos de la aplicación indican que será necesario usar pantallas stateless y statefull, donde pueda variar o no el contenido (los widgets) de estas. La pantalla principal donde se podrán ver las notas será statefull, en cambio, la pantalla de creación de la nota (el formulario) sería stateless.

Definir la arquitectura del sistema

Selección de Patrones de Arquitectura

Primero seleccionaremos el patrón Provider para manejar el estado y aspecto de la aplicación. Lo elegimos ya que es el adecuado para gestionar aplicaciones de tamaño medio o pequeño, y por ahora no es una aplicación de gran tamaño. Además, el patrón provider permite que tengamos acceso a datos de la aplicación a través del uso de "state =".

Evaluación de Tecnologías

El código de la aplicación se escribirá en Flutter, con el lenguaje Dart. Se añadirán librerías (a través del archivo "pubspec.yaml") para implementar códigos auxiliares para agilizar la programación, que se obtendrán principalmente de la plataforma "pub.dev". En cuanto al almacenamiento de datos, se conectará una base de datos con MySQL.

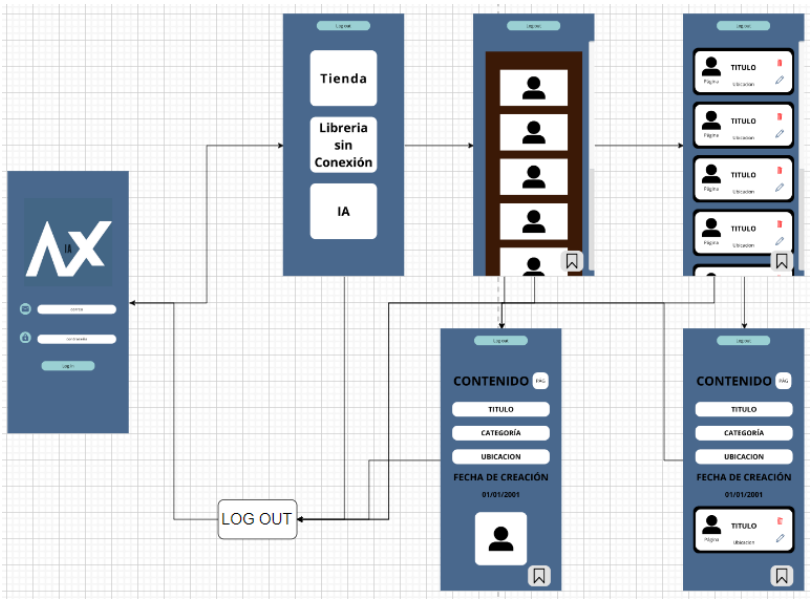
Definición de la Infraestructura

La infraestructura se definirá en varios apartados:

- Frontend: Se usará Flutter (en Dart) para la programación de la aplicación y su interfaz.
- Backend: Se procesan las solicitudes del usuario y se conectará una base de datos SQL (con SQLite).

Diseño de interfaces

Prototipado y Wireframes



Principios de UX

La aplicación se diseñará de modo que el usuario pueda realizar las acciones que quiera con facilidad. Se programará para que sea accesible desde cualquier dispositivo, adaptándose a diferentes resoluciones de pantalla y dispositivos. En próximas actualizaciones se implementarán funciones de accesibilidad para usuarios que no puedan usar la aplicación con las funciones básicas, como usuarios con limitaciones físicas.

Diseño de datos

Modelado de Datos

Los datos de la aplicación se guardarán y gestionarán a través de una base de datos no relacional creada en MySQL. Esta incluirá tablas como datos_usuario, libros_virtuales,bibliotecas_virtuales y categorías.

Normalización y Optimización

Se requiere una base de datos funcional y eficiente, por lo que se aplicarán principios de normalización para organizar y transformar la información para evitar redundancias ineficientes. Siempre se intentará optimizar las consultas al máximo escribiéndolas de la forma más eficiente (con SQL).

Consideraciones de seguridad y rendimiento

Análisis de riesgos

Es posible que la base de datos sufra algunos ciberataques que podrían crear riesgos de pérdidas de datos o manipulación indebida de los mismos. Los datos de cada usuario en la aplicación no son relevantes para un atacante, pero si los datos personales del propio usuario.

Autenticación y autorización

Se va a implementar un sistema de verificación en 2 pasos con métodos de autenticación como un segundo correo electrónico o un número de teléfono. Se podrán modificar los ajustes de la aplicación para que el usuario establezca cuando quiere que le pida verificar su identidad.

Encriptación de datos

Se cifrarán los datos de usuario a través de protocolos como TLS (Seguridad de la capa de transporte) y HTTPS (Protocolo seguro de transferencia de hipertexto).

Seguridad en el diseño de la base de datos

Las bases de datos se crearán específicamente para que solo los usuarios con el rol de administrador puedan acceder a los datos almacenados. Se usarán vistas reducidas para cada usuario (para que acceda a sus datos con facilidad y de forma segura).

Prevención de inyecciones

Se establecerán limitaciones en los tipos de operaciones que los usuarios pueden realizar en la base de datos para mitigar posibles ataques o acciones no deseadas que pudieran comprometer la seguridad o la integridad de los datos. Estas limitaciones pueden ser la aplicación de declaraciones preparadas y consultas parametrizadas.

Gestión de errores y registros

Se creará un sistema de registro para guardar errores sin la posibilidad de fugas de datos. Para ello, se guardarán los datos en servidores con seguridad alta y con acceso restringido. Además, cuando se detecte algún tipo de anomalía o incidente, se mandará una notificación de alerta a los administradores.

Conclusión

Alexandr.IA es una idea revolucionaria para la organización de librerías, que proporciona al lector todas las herramientas y soluciones para mejorar su lectura. Creo que los lectores deberían empezar a usar las nuevas tecnologías para optimizar su tiempo y aprovechar así mejor sus tiempos de lectura. En mi caso, todo ha sido mucho más cómodo desde que uso Alexandr.IA.

Bibliografía

Investigación de software y estudio de viabilidad

Análisis de mercado:
https://www.adslzone.net/noticias/moviles/top-apps-control-libros/

Evaluación técnica:
https://www.linkedin.com/pulse/7-aspectos-t%C3%A9cnicos-que-debes-conocer-antes-de-crear-una-pintos/

https://www.grupoebim.com/blog/ventajas-desventajas-flutter/

Viabilidad económica:

https://cloud.google.com/learn/what-is-paas?hl=es

https://copimar.net/mantenimiento-informatico-tarifa-plana/

https://www.capterra.es/directory/32376/platform-as-a-service/pricing/free/software

Estudio de viabilidad, marco TELOS – Operacional:

https://asana.com/es/resources/implementation-plan

Cascada Etapa 1: Requisitos y documentación

Documentación de Flutter

Requisitos:

https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-y-aplicaciones/

Modelado de datos:

https://apiumhub.com/es/tech-blog-barcelona/elementos-ventajas-desarrollo-basado-en-datos/

Máquinas de estado:

https://www.profesionalreview.com/2022/11/26/maquinas-de-estado-finito-que-son-para-que-sirven/

Diagrama de casos de uso:

https://creately.com/blog/es/diagramas/tutorial-diagrama-caso-de-uso/

Diagrama de flujo:

https://miro.com/es/diagrama-de-flujo/que-es-diagrama-de-flujo/

Diagrama de estado:

https://www.edrawsoft.com/es/uml/uml-state-diagram.html

Diseño

Patrón Provider:

https://www.patterns.dev/vanilla/provider-pattern/

UX:

https://platzi.com/blog/principios-basicos-ux-design/

Definición de la infraestructura:

https://media.licdn.com/dms/image/D4D10AQH6d6xJ87KOCA/image-shrink_800/0/1702566804119?e=1713859200&v=beta&t=OmfrHkaVuSGeHpTATehMWa4s74yBrJ8pmkG9ezWUujk

Encriptación de datos:

https://www.icm.es/2022/05/18/diferencias-entre-protocolos-ssl-tls-y-https/

Prevención de inyecciones:

https://www.cloudflare.com/es-es/learning/security/threats/how-to-prevent-sql-injection/

Código

El código de Alexandr.IA se encuentra en GitHub en la rama “master”:

https://github.com/mesi260/Alexandr_IA.git