**1. Uso de la Base de Datos**

El sistema utiliza una **base de datos relacional** (ej. PostgreSQL o MySQL) porque:

* **Relaciones claras y estructuradas**: se deben manejar usuarios, roles, proyectos, fases y documentos. Estos datos están interconectados (ejemplo: un usuario puede estar en varios proyectos y un proyecto puede tener varios documentos). Una base relacional permite modelar estas relaciones de forma eficiente.
* **Trazabilidad y control de versiones**: la BD guarda historial de documentos, privilegios y fases del proyecto, lo que asegura un seguimiento completo.
* **Consistencia de datos**: al ser un sistema que maneja información sensible (usuarios, permisos, auditorías), se necesita integridad referencial y reglas claras de validación.

👉 En el **diagrama de BD** se ven entidades como *Usuario, Proyecto, Documento, Rol, Fase, Privilegio*, conectadas con relaciones uno-a-muchos y muchos-a-muchos. Esto refleja cómo la herramienta gestiona de manera organizada la documentación y las interacciones entre personas y proyectos.

**2. Uso del Diagrama de Arquitectura**

El diagrama muestra una **arquitectura en la nube (AWS)** con separación de componentes:

* **Login + 2FA**: asegura que solo usuarios autorizados ingresen.
* **Load Balancer (ELB)**: distribuye tráfico y garantiza alta disponibilidad.
* **VPC con subred pública/privada**: lo expuesto (ej. acceso web) se separa de lo interno (app y BD) para mayor seguridad.
* **Aplicación central en capa privada**: allí corre DocuFlow, protegido de accesos externos.
* **Amazon RDS + S3**: permiten manejar datos estructurados (relacional) y documentos (archivos).
* **Servicios de seguridad (IAM, WAF, HTTPS/TLS, encriptación AES-256)**: evitan accesos indebidos y ataques.
* **API**: habilita integración futura con otros sistemas.
* **IA (futura)**: pensada como una mejora para generar y revisar documentos automáticamente.

👉 En resumen, la arquitectura asegura **seguridad, escalabilidad y disponibilidad** para el sistema.

**3. Por qué convienen estas herramientas para el proyecto**

* **Bases de datos relacionales (PostgreSQL/MySQL):** robustas, seguras y bien soportadas para proyectos académicos y empresariales.
* **NoSQL (DynamoDB/Firestore, opcional):** útil para logs y auditorías de alto volumen, lo que descarga a la BD principal.
* **Despliegue en la nube (AWS/GCP/Azure):** asegura accesibilidad global, crecimiento escalable y redundancia en caso de fallas.
* **Arquitectura en 3 capas (presentación, lógica de negocio, datos):** separa responsabilidades, facilitando el mantenimiento, la prueba y futuras mejoras.
* **Python/Django:** framework rápido y seguro para backend, con librerías listas para roles, seguridad y APIs.
* **HTML/CSS/JS en frontend:** garantizan compatibilidad con cualquier navegador y facilidad de uso.
* **Seguridad avanzada (IAM, WAF, TLS):** clave porque el sistema maneja documentos internos y permisos diferenciados.

✅ En pocas palabras:  
La base de datos se usa porque permite estructurar, relacionar y dar trazabilidad a la información crítica. La arquitectura en la nube y de tres capas garantiza que DocuFlow sea seguro, escalable y fácil de mantener. Y las tecnologías elegidas (PostgreSQL/MySQL, Python/Django, AWS) convienen porque son estándares industriales probados, flexibles y adecuados tanto para el prototipo académico como para un futuro despliegue empresarial.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_0\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**🗃️ ¿Por qué se utiliza esa base de datos?**

El modelo de base de datos propuesto en DocuFlow está diseñado para reflejar relaciones reales entre entidades clave del proyecto, como usuarios, empresas, documentos y fases. Se utiliza una base de datos relacional (PostgreSQL o MySQL) por varias razones:

* **Estructura clara y normalizada**: Las tablas como País, Región, Comuna, Empresa, Proyecto, Usuario, Documento, etc., están organizadas jerárquicamente y con claves foráneas que aseguran integridad referencial.
* **Trazabilidad y control**: La relación entre documentos y fases del proyecto (Fase\_Documento), y entre usuarios y proyectos (Usuario\_Proyecto), permite saber quién hizo qué, cuándo y en qué contexto.
* **Gestión de permisos**: La tabla Privilegio permite definir qué acciones puede realizar cada usuario sobre cada documento, lo que es esencial para la seguridad y el flujo de trabajo.
* **Escalabilidad**: Al estar bien normalizada, la base puede crecer sin volverse caótica, manteniendo el orden y la eficiencia.

En resumen, se usa una base relacional porque el modelo requiere manejar datos estructurados, relaciones complejas y trazabilidad precisa, todo lo cual es ideal para este tipo de tecnología.

**📊 ¿Por qué se organiza así el diagrama de base de datos?**

El diagrama refleja una lógica de negocio clara y modular:

* **Separación por niveles geográficos** (País → Región → Comuna) permite contextualizar usuarios y empresas.
* **Usuarios y roles** están separados en Usuario, Perfil y Rol, lo que facilita la gestión de permisos y personalización.
* **Documentos y fases** están conectados por Fase\_Documento, lo que permite saber en qué etapa se encuentra cada archivo.
* **Privilegios** se asignan por documento y usuario, lo que da flexibilidad y seguridad.

Esta estructura permite que el sistema sea adaptable a distintos contextos organizacionales, manteniendo la trazabilidad y el control.

**🏗️ ¿Cómo funciona la arquitectura del sistema?**

La arquitectura es de tres capas, desplegada en la nube (AWS, GCP o Azure):

1. **Capa de Presentación (Frontend)**:
   * HTML, CSS y JavaScript.
   * Muestra la interfaz al usuario y envía solicitudes al backend.
2. **Capa de Lógica de Negocio (Backend)**:
   * Python con Django.
   * Gestiona usuarios, permisos, flujos de trabajo y validaciones.
   * Expone una API que conecta con el frontend.
3. **Capa de Datos**:
   * Bases relacionales para datos estructurados (usuarios, proyectos, documentos).
   * Bases NoSQL para logs y auditoría (alta velocidad de escritura).

Además, se despliega en la nube con componentes como:

* **Elastic Load Balancer**: distribuye tráfico.
* **VPC con subredes públicas y privadas**: separa lo expuesto de lo interno.
* **Amazon RDS y S3**: para datos estructurados y archivos.
* **IAM y WAF**: para seguridad y control de accesos.

**🧰 ¿Por qué convienen estas herramientas?**

* **Escalabilidad**: La arquitectura permite crecer sin rediseñar todo.
* **Seguridad**: IAM, WAF, TLS y encriptación AES-256 protegen los datos.
* **Trazabilidad**: Logs, historial de versiones y privilegios permiten saber quién hizo qué.
* **Flexibilidad**: El sistema puede adaptarse a cualquier empresa gracias a los mantenedores.
* **Colaboración**: Herramientas como Confluence y Notion permiten trabajo en equipo.
* **Portabilidad**: El enfoque multi-cloud permite desplegar en cualquier proveedor.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_0\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

¡Entendido! Aquí tienes la explicación del proyecto DocuFlow, sus decisiones tecnológicas y la arquitectura, presentada en un formato fluido y sin cuadros comparativos.

🎯 Propósito y Contexto del Proyecto DocuFlow

DocuFlow nace como una solución a una problemática crítica en el desarrollo de

*software*: la ausencia de un proceso claro y estandarizado para la creación y mantención de la documentación técnica.

Este proyecto se concibe como un

modelo de gestión y estandarización diseñado para optimizar los procesos de documentación y gestión de proyectos. Su meta es garantizar orden, claridad y continuidad en los sistemas desarrollados. Si bien DocuFlow es escalable a cualquier industria , el alcance del proyecto académico se centra en la implementación y validación de un

prototipo funcional en un entorno simulado de Duoc UC.

🏗️ Arquitectura y Diseño Lógico del Sistema

El proyecto DocuFlow se sustenta en una sólida

arquitectura de tres capas. Este modelo fue elegido por su capacidad de

separar claramente las responsabilidades de cada componente, lo que facilita enormemente el desarrollo, la mantención y la escalabilidad futura del sistema.

1. Capa de Presentación (Frontend): Es la interfaz con la que el usuario interactúa en el navegador. Se desarrolla con tecnologías web estándar como

HTML, CSS y JavaScript para asegurar la compatibilidad y accesibilidad.

1. Capa de Lógica de Negocio (Backend): Actúa como el cerebro de la aplicación. Aquí residen todas las reglas de negocio, la gestión de permisos, los flujos de trabajo (aprobación y revisión) y la futura integración con herramientas de IA. Se construye con

Python/Django, exponiendo una API que la capa de presentación consume.

1. Capa de Datos: Es la responsable de almacenar y gestionar toda la información. Se utiliza un enfoque mixto de bases de datos para optimizar el rendimiento y la escalabilidad.

💾 La Elección de la Base de Datos y su Modelo

La estrategia de DocuFlow es utilizar dos tipos de bases de datos, lo que resulta conveniente para manejar la complejidad y el volumen de la información del proyecto:

* Bases de Datos Relacionales (PostgreSQL / MySQL): Se escogieron por su fiabilidad, robustez y madurez. Son ideales para la gestión de

datos estructurados y las relaciones complejas, como los usuarios, sus permisos y la estructura de los proyectos.

* Bases de Datos NoSQL (DynamoDB / Firestore): Se utilizarán para manejar datos no estructurados, específicamente los logs de actividad y los eventos de auditoría. Son convenientes en este caso porque requieren una

alta velocidad de escritura y una escalabilidad masiva para el registro continuo de actividad.

El

Diagrama de Base de Datos se utiliza para modelar esta estructura de manera visual y clara. Muestra las entidades fundamentales del sistema, como

ADMIN\_USUARIO, ADMIN\_PROYECTO, ADMIN\_DOCUMENTO y ADMIN\_ROL, y establece las conexiones lógicas entre ellas, siendo esencial para definir el correcto funcionamiento de los permisos y la trazabilidad documental.

☁️ Explicación del Diagrama de Arquitectura (Despliegue en la Nube)

El diagrama de arquitectura representa el despliegue del sistema DocuFlow en la nube, específicamente en

Amazon Web Services (AWS). Esta infraestructura se elige porque garantiza que el sistema sea accesible desde cualquier lugar, tenga

alta disponibilidad y pueda escalar dinámicamente.

Los componentes clave y su conveniencia son:

* Seguridad y Acceso (Login + 2FA, AWS IAM): Se implementa la autenticación de doble factor para garantizar el acceso solo a usuarios autorizados.

AWS IAM (Identity and Access Management) es conveniente para controlar los permisos y roles dentro del entorno AWS, asegurando que cada usuario tenga solo lo que le corresponde.

* Infraestructura Segura (VPC, ELB): Se utiliza una VPC (Nube Privada Virtual) con subredes pública y privada para separar los componentes expuestos a internet (como el Elastic Load Balancer - ELB) de los internos (la aplicación DocuFlow y la base de datos). El

ELB distribuye el tráfico, asegurando alta disponibilidad y evitando sobrecargas.

* Protección (AWS WAF): El AWS WAF (Web Application Firewall) es conveniente para proteger la aplicación contra ataques web comunes y tráfico malicioso, como inyecciones SQL o *scripts* maliciosos.
* Almacenamiento (Amazon RDS y S3): Amazon RDS aloja la base de datos relacional (datos estructurados), y un Storage Bucket (S3) actúa como repositorio de archivos (documentos, imágenes) con alta escalabilidad y seguridad.
* Integración y Trazabilidad: El uso de una API permite que otros sistemas se comuniquen con DocuFlow, facilitando la integración. Además, se utiliza

Amazon CloudWatch para el registro de auditoría, esencial para la trazabilidad completa de cada acción realizada en el sistema.

¡Claro! Aquí está la explicación del proyecto DocuFlow, sus decisiones de arquitectura y base de datos, presentada en formato de texto listo para exposición y sin cuadros comparativos.

**🏗️ Explicación del Diagrama de Arquitectura**

La arquitectura de DocuFlow se basa en un robusto diseño de **tres capas** (Presentación, Lógica de Negocio y Datos), desplegado sobre una **infraestructura en la nube** (AWS, GCP o Azure). Esta elección es fundamental para asegurar **escalabilidad, disponibilidad y seguridad** del sistema.

La entrada al sistema está protegida con **Login y Doble Factor de Autenticación (2FA)**. El tráfico se canaliza a través de un **Load Balancer (ELB)**, que se encarga de distribuirlo entre los servidores, garantizando la **alta disponibilidad** y evitando sobrecargas.

La capa de infraestructura utiliza una **VPC (Nube Privada Virtual)**, esencial para la seguridad: los componentes expuestos a internet (como el acceso web) se ubican en una subred pública, mientras que la aplicación central (**DocuFlow**) y las bases de datos residen en una **subred privada**, protegidas de accesos directos.

Los componentes de seguridad son vitales: **AWS IAM** controla estrictamente los roles y permisos dentro del entorno de nube, y el **AWS WAF (Web Application Firewall)** protege la aplicación central contra ataques comunes. La **API** se expone para habilitar una integración futura con otros sistemas. Finalmente, la arquitectura ya contempla la integración de **Inteligencia Artificial (IA)** para generar y revisar documentos automáticamente, marcando una ruta de mejora a futuro.

**💾 Uso y Explicación del Diagrama de Base de Datos**

El sistema utiliza una base de datos principalmente **relacional** (como PostgreSQL o MySQL). Esta elección es conveniente porque los datos del proyecto (usuarios, roles, proyectos, fases y documentos) están **interconectados** y requieren **relaciones claras y estructuradas** para su manejo eficiente.

La base relacional es esencial para:

1. **Garantizar Trazabilidad y Control:** Almacena el historial de documentos, versiones, privilegios y fases, asegurando un seguimiento completo de todas las acciones.
2. **Asegurar Consistencia:** Permite aplicar integridad referencial y reglas de validación necesarias para manejar información sensible, como permisos y auditorías.

El **Diagrama de Base de Datos** refleja esta lógica de negocio con una estructura clara y normalizada. Se observan entidades clave como **Usuario, Proyecto, Documento, Rol, Fase y Privilegio**, todas interconectadas mediante relaciones uno-a-muchos y muchos-a-muchos. Esta organización modular permite la **gestión precisa de permisos** y el seguimiento de cada documento a lo largo de las diferentes etapas del proyecto. Opcionalmente, se plantea usar bases **NoSQL** (como DynamoDB) para manejar logs de auditoría de alto volumen, descargando el trabajo de la base de datos principal.

**✅ Por qué Convienen estas Herramientas**

Las tecnologías y el diseño de DocuFlow convienen porque:

* **Escalabilidad y Disponibilidad:** La arquitectura en la nube (AWS/GCP/Azure) garantiza **accesibilidad global, crecimiento escalable** y redundancia en caso de fallas.
* **Seguridad:** Servicios como **IAM, WAF, TLS** y la encriptación de datos protegen el sistema, lo cual es clave dado que se gestiona documentación interna y permisos diferenciados.
* **Desarrollo Rápido y Robusto:** El uso de **Python/Django** es ideal para el *backend* por su velocidad y seguridad, con librerías que facilitan la gestión de roles, APIs y seguridad.
* **Mantenibilidad:** La **arquitectura de tres capas** separa las responsabilidades, facilitando el mantenimiento, las pruebas y las futuras mejoras del sistema.

En síntesis, la combinación de una base de datos estructurada para la trazabilidad y una arquitectura en la nube segura hace de DocuFlow una solución **robusta, flexible y adecuada** tanto para un prototipo académico como para un despliegue empresarial.

Gary, dice mucho “por lo cual”