Arquitectura Via Alta

Vía Alta es una aplicación web en desarrollo para el Instituto Via Diseño, con la intención de generar horarios de clases de manera semi-automática, con el objetivo de centralizar la gestión de los horarios e inscripciones, reducir el tiempo de creación de horarios para las coordinadoras, reducir el tiempo de inscripción para los alumnos, minimizar los errores de transcripción en los registros de inscripción, y optimizar la disponibilidad de aulas y profesores antes del próximo período de inscripciones.

Las coordinadoras deberán poder ingresar la disponibilidad de horario de los profesores, así como limitantes y reglas relacionadas a la creación de horarios. Las coordinadoras podrán generar turnos de horarios, consultar, modificar, guardar y eliminar horarios por alumnos individuales, entre otras funcionalidades.

Por su parte, los alumnos podrán ingresar a la aplicación durante su turno de inscripción, consultar su propuesta de horario, y registrar aceptación o solicitar cambios al mismo.

Dadas las características mencionadas, así como todos los requisitos establecidos para la aplicación, se ha decidido utilizar las siguientes tecnologías para la realización del proyecto:

Front-end	**Back-end**
Tailwind CSS React.js	TypeScript Next.js PostgreSQL

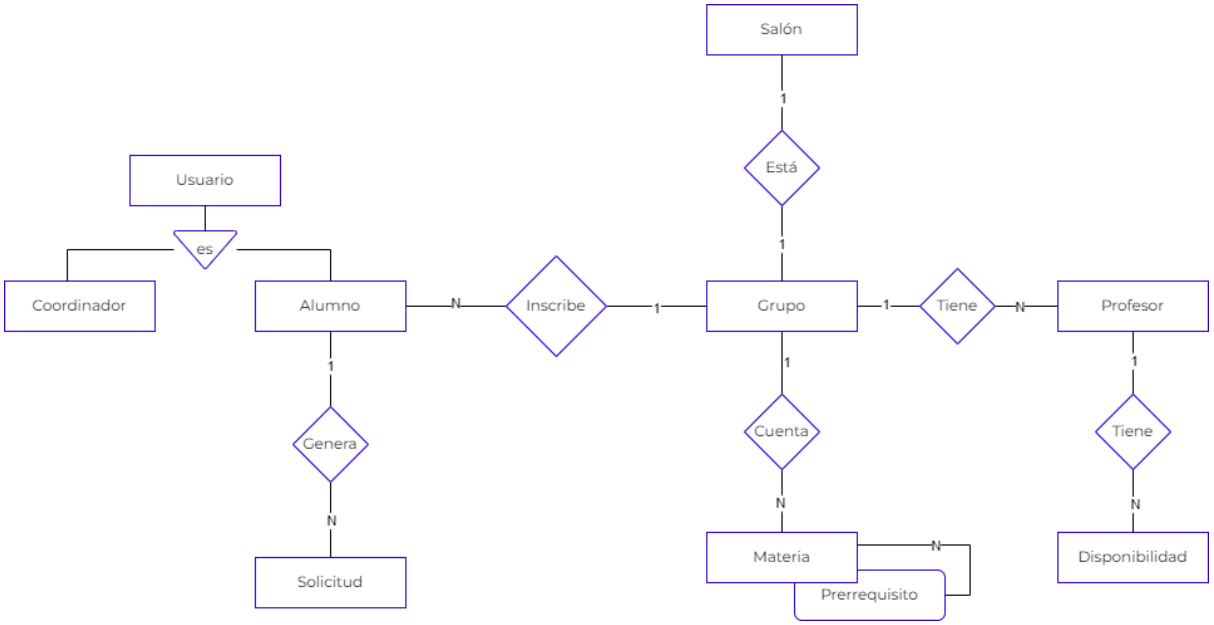
La aplicación será desarrollada siguiendo la arquitectura *Modelo Vista Controlador* (MVC), con la intención de promover buenas prácticas de desarrollo, seguimiento de patrones de diseño (singleton, etc), mantener cohesión entre diagramas de secuencia, y diseñar la aplicación de manera consciente.

Siguiendo esta lógica, el directorio de la aplicación deberá lucir de la siguiente manera:

```
Unset
/via-alta
|-- /components            #componentes reutilizables (interfaces)
|   |-- Button.tsx
|   |-- Navbar.tsx
|   |-- Card.tsx
|   |-- Component.tsx
|-- /controllers           #intermediario entre vistas y modelo de datos
|   |-- postController.ts
|   |-- controller.ts
|-- /models                #lógica de los datos
|   |-- userModel.ts
|   |-- postModel.ts
|-- /pages                 #vistas
|   |-- /api               #rutas
|   |   |-- user.ts        #endpoint (lógica de controlador)
|   |   |-- post.ts        #endpoint (lógica de controlador)
|   |-- index.tsx          #vista
|   |-- /users
|   |   |-- index.tsx
|   |   |-- [id].tsx       #vista por detalle
|   |-- /posts
|   |   |-- index.tsx
|   |   |-- [id].tsx
|-- /public
|   |-- /images
|   |-- /fonts
|-- /styles
|   |-- globals.css        #estilos globales
|   |-- tailwind.css
|-- /lib                   #equivalente a utils
|   |-- db.ts              #conexión con Postgress
|   |-- prisma.ts          #conexión usando prisma
|-- /types                 #types - definen interfaces de datos
|   |-- user.d.ts
|   |-- post.d.ts
|-- /node_modules          #dependencias
|-- .env                   #variables de ambiente
|-- next.config.js         #configuración Next.js
|-- tailwind.config.js     #configuración de Tailwind CSS
|-- tsconfig.json          #configuración de TypeScript
|-- package.json           #definición de dependencias
|-- README.md              #documentación
```

Esta arquitectura y stack tecnológico permitirán cumplir con los requisitos no funcionales establecidos de rendimiento, escalabilidad, disponibilidad, seguridad, compatibilidad, mantenibilidad, usabilidad y confiabilidad.

Así mismo, la aplicación deberá poder comunicarse con la base de datos del sistema central de administración del Instituto Via Diseño por medio del protocolo http, de tal forma que el sistema de inscripciones pueda minimizar el número de tablas y registros, así como mantener referencia a cualquier cambio desde el sistema central.
La aplicación seguirá el siguiente modelo de datos:



Rutas:

En Next.js, las rutas se basan en el sistema de archivos, lo que significa que cada archivo dentro de la carpeta pages (en el ejemplo de la estructura de archivos corresponde a una ruta en la aplicación), es una ruta.

Agregando el periodo académico a la ruta, se podrá visualizar la información del periodo académico, que es uno de los requisitos funcionales.

- /[periodo-academico]/horario-general ->coordinador genera horario general y coordinador modifica horario y coordinador guarda horario

- /[periodo-academico]/horarios/[semestre-id] -> coordinador consulta horario por semestre
- /[periodo-academico]/horarios/[estatus] -> coordinador consulta estatus de inscripción de alumnos
- /[periodo-academico]/horarios/turnos -> coordinador genera turnos de inscripción
- /profesores -> coordinador gestiona disponibilidad de profesores
- /salones -> coordinador gestiona disponibilidad de salons
- /[periodo-academico]/turnos/[alumno] -> alumno consulta turno de inscripción y coordinador consulta turno de inscripción por alumno
- /[periodo-academico]/[horario-alumno] -> alumno consulta propuesta de horario y alumno acepta/solicita cambios a propuesta de horario y coordinador modifica horario individual por alumno