Tecnológico de Monterrey Campus Santa Fe

Documentación: TecGPT

Alumnos:

Dulce Daniela García Ruiz A01748013

Paula Verdugo Márquez A01026218

Miguel Ángel Cabrera Victoria A01782982

Ian Luis Vázquez Morán A01027225

Profesores:

Jorge Rodríguez Ruiz

Octavio Navarro Hinojosa

Junio 6, 2025

Tabla de Contenidos

Documentación: TecGPT

Introducción

TecGPT nace de la necesidad de optimizar la atención a consultas administrativas básicas en el Tecnológico de Monterrey Campus Santa Fe. Actualmente, estudiantes y personal generan múltiples consultas de soporte para preguntas simples como "¿Dónde saco mi credencial?" o "¿Cuándo es el último día para pagar la colegiatura?". TecGPT centraliza estas respuestas comunes en una interfaz conversacional, liberando al personal para atender casos más complejos que requieren intervención humana. El proyecto busca implementar una solución de inteligencia artificial que proporcione respuestas inmediatas a estas consultas frecuentes, liberando recursos del personal administrativo para atender asuntos más complejos.

El objetivo del proyecto es desarrollar un chat de inteligencia artificial que proporcione respuestas inmediatas a consultas administrativas frecuentes del campus, reduciendo la necesidad de contactar directamente al personal o realizar búsquedas extensas. Los objetivos específicos incluyen crear un modelo de lenguaje personalizado entrenado con alrededor de 700 preguntas específicas del Tecnológico de Monterrey, implementar una aplicación web accesible que permita consultas, desarrollar un sistema de autenticación, y establecer una base tecnológica escalable para futuras expansiones del sistema.

Requisitos del Sistema

Requisitos Funcionales

Gestión de usuarios: El sistema permite registro de nuevos usuarios con validación de email institucional @tec.mx, autenticación mediante credenciales, autorización basada en tokens JWT con expiración y cierre de sesión seguro.

Sistema de chat: La aplicación proporciona interfaz de chat en tiempo real para consultas con recepción y visualización de respuestas del modelo, mantenimiento de historial durante la sesión e indicadores visuales de estado cuando el prompt este cargando.

Procesamiento de IA: El sistema integra un modelo de lenguaje personalizado que procesa consultas administrativas del Tec y genera respuestas coherentes basadas en dataset de más de 700 preguntas con tiempo de procesamiento menor a 15 segundos.

Gestión de sesiones: La aplicación crea sesiones de usuario al autenticarse, incluye logging de eventos de autenticación y autorización, con renovación automática de tokens y terminación de sesiones inactivas.

Interfaz de usuario: El frontend implementa diseño responsivo para móviles y escritorio, tema claro y oscuro con persistencia y manejo de estados de carga.

Validación y seguridad: El sistema encripta las contraseñas con bcrypt, valida que el formato del email sea institucional, aplica políticas de contraseñas seguras al registrar un usuario y limita las rate limiting de 100 peticiones por 15 minutos.

Rendimiento: La aplicación garantiza tiempo de respuesta del chat menor a 15 segundos y carga inicial de página menor a 2 segundos.

Seguridad: El sistema utiliza cifrado bcrypt para contraseñas, tokens JWT con expiración, HTTPS obligatorio y rate limiting implementado.

Escalabilidad: La arquitectura modular está preparada para crecimiento con contenedores Docker implementados para escalamiento futuro y base de datos optimizada.

Usabilidad: La interfaz es intuitiva incluso para usuarios nuevos.

Accesibilidad: El sistema cumple estándares de contraste de colores, navegación por teclado completa, modo claro y oscuro para reducir fatiga visual y tamaños de fuente configurables.

Mantenibilidad: El código está documentado con arquitectura separada en capas y logs detallados para debugging y mantenimiento futuro.

Casos de Uso Principales

Componente	Caso de Uso	Descripción	Flujo
Cliente/Front-End	CU-01: Registro de nuevo usuario	Estudiante del Tec crea cuenta nueva en TecGPT para acceder al chatbot	Acceso a registro, completar formulario con datos institucionales, validación formato @tec.mx, redirección automática a interfaz de chat
Cliente/Front-End	CU-02: Inicio de sesión de usuario existente	Usuario con cuenta válida accede al sistema para usar el chatbot	Página de login, ingreso de credenciales, validación contra backend, token almacenado en localStorage, acceso a chat
Cliente/Front-End	CU-03: Realizar consulta administrativa al chatbot	Usuario autenticado hace preguntas sobre servicios del campus	Escribir pregunta, envío al backend con token, procesamiento por IA, mostrar respuesta en interfaz
Cliente/Front-End	CU-04: Cambiar tema visual de la aplicación	Usuario personaliza la interfaz según su preferencia visual	Selección entre modo claro/oscuro, aplicar estilos CSS correspondientes, persistir preferencia en localStorage
Servidor/Back-End	CU-05: Autenticar credenciales de usuario	Sistema valida identidad de usuario que intenta acceder	Recibir datos de login, verificar formato @tec.mx, comparar contraseña con hash bcrypt, generar y retornar JWT
Servidor/Back-End	CU-06: Crear nuevo usuario en el sistema	Sistema registra estudiante nuevo con validaciones de seguridad	Validar unicidad de email, aplicar hash bcrypt a contraseña, almacenar en base de datos, asignar rol USER
Servidor/Back-End	CU-07: Procesar consulta de inteligencia artificial	Sistema maneja pregunta del usuario y obtiene respuesta de IA	Validar token de autorización, reenviar consulta a modelo externo, recibir respuesta, retornar JSON estructurado

Base de Datos	CU-08: Gestionar información de usuarios registrados	Sistema almacena y mantiene datos de estudiantes del Tec	Operaciones CRUD sobre tabla users con campus, nombre, email y contraseña hasheada
Base de Datos	CU-09: Registrar actividades del sistema para auditoría	Sistema mantiene historial de acciones para seguridad y análisis	Insertar registros en tabla logs con intentos de autenticación y acciones de usuario
Servicios Externos	CU-10: Obtener respuesta de modelo de IA externo	Sistema consulta servicio de IA para responder preguntas administrativas	Endpoint /conversation, comunicación con servicio externo, procesamiento de respuesta, retorno al frontend

Descripción de la aplicación de IA

TecGPT utiliza inteligencia artificial para automatizar respuestas a consultas administrativas frecuentes del Tecnológico de Monterrey Campus. La IA funciona como primera línea de atención, proporcionando información inmediata sobre servicios del campus y liberando al personal administrativo para casos más complejos.

El sistema emplea la arquitectura tradicional de un GPT compuesto de varias cabeceras de atención. El modelo fue entrenado con aproximadamente 700 preguntas administrativas específicas del Tec. La implementación en inglés optimiza el procesamiento y mejora la precisión de respuestas, ya que en el lenguaje español hay diversas palabras que tienen diferentes significados que podrían afectar la coherencia del texto.

TecGPT aborda la saturación de consultas administrativas básicas que consumen tiempo del personal y generan esperas innecesarias para estudiantes. El sistema centraliza información dispersa sobre ubicaciones, horarios, fechas límite y procedimientos, proporcionando acceso inmediato sin depender de atención presencial o búsquedas extensas en sitios web institucionales.

El desarrollo de un modelo propio garantiza especialización en el contexto específico del Tecnológico de Monterrey, privacidad total de datos institucionales y control completo sobre la calidad de respuestas. Esta aproximación elimina dependencias de servicios externos, reduce costos operativos a largo plazo y permite personalización según terminología y procedimientos específicos del campus.

Arquitectura del Sistema

Arquitectura de la Aplicación

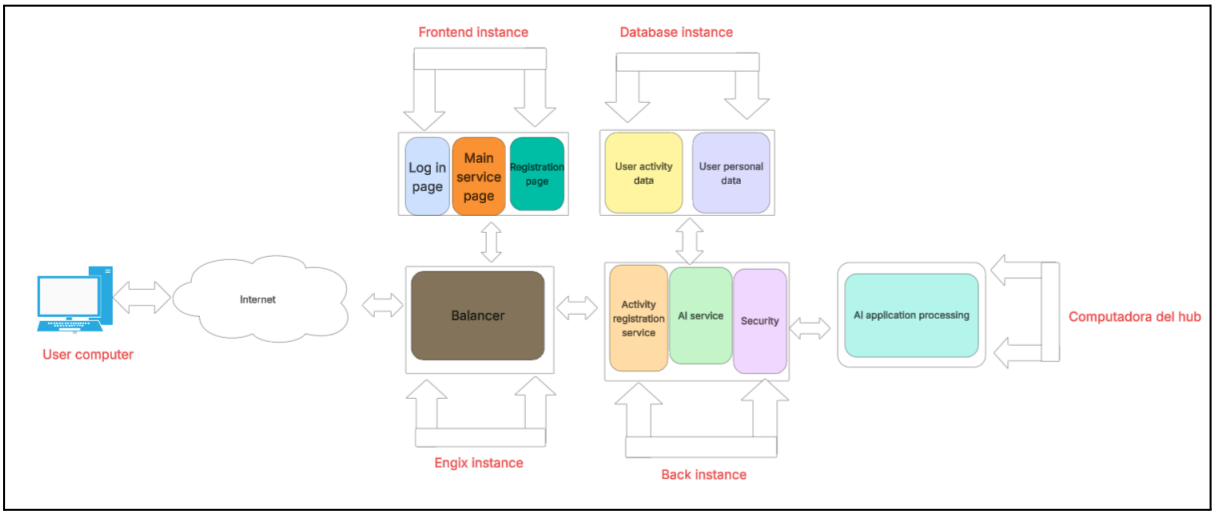


Imagen 1. Creado en: LucidChart.

Esta imagen representa la arquitectura interna del proyecto y las diferentes instancias que lo componen, distribuidas de manera modular para facilitar su entendimiento. A continuación, se describen las funciones y conexiones de cada parte del sistema:

User Computer

- Es el punto de entrada al sistema.
- El usuario accede a la aplicación a través de su computadora utilizando Internet.

Engix Instance (Instancia del Balanceador)

- Balancer:
- Se encarga de distribuir las solicitudes del usuario y del sistema entre las distintas instancias con sus respectivos servicios

Frontend Instance (Instancia de Frontend)

Log in page:

- Página de inicio de sesión para autenticar al usuario.

Registration page:

- Página para registrar una nueva cuenta.

Main service page:

- Página principal donde el usuario accede al servicio de IA

Back Instance (Instancia de Backend)

Activity registration service:

- Se encarga de registrar las actividades del usuario.

AI service:

- Módulo que se encarga de mandar la solicitud del usuario a la computadora del hub y mandar la respuesta correspondiente a la instancia de Engix Instance

Security:

- Encargado de verificar la identidad, manejar permisos y proteger la información.

Database Instance (Instancia de Base de Datos)

User activity data:

- Datos relacionados con las acciones del usuario.como solicitudes a la AI y sus respectivos resultados

User personal data:

- Información personal del usuario como nombre, campus, etc.

Computadora del Hub (Hub Computer)

AI application processing:

- Procesa las solicitudes del usuario con algoritmos de AI y se conecta con el AI service del backend para entregar resultados procesados.

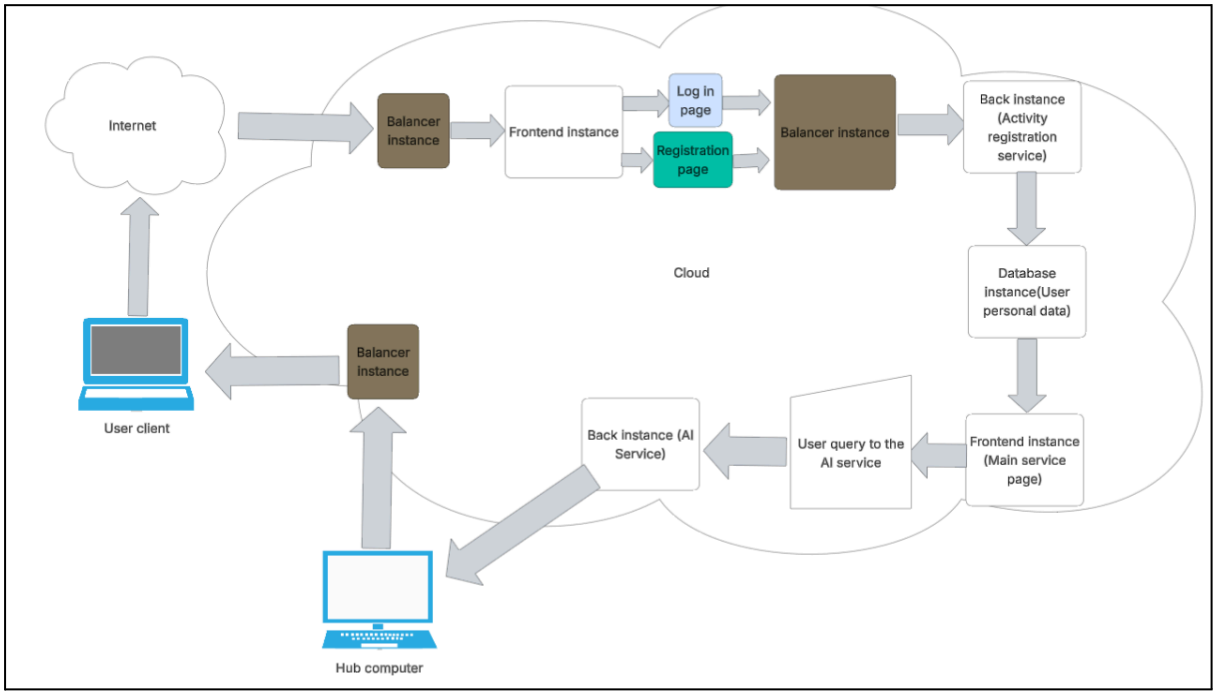


Imagen 2. Creado en: LucidChart.

Este segundo diagrama representa la interacción del usuario con las distintas funcionalidades del sistema, manteniendo la lógica descrita anteriormente (uso de balanceadores, Engix y estructura en la nube), pero centrándose ahora en los flujos funcionales entre páginas, servicios y la base de datos.

Interacción del Usuario

El usuario cliente accede a la aplicación desde Internet y es dirigido inicialmente a la instancia de frontend, donde puede ingresar a:

Log in page (para autenticación).

Registration page (para crear una cuenta nueva).

Registro y Autenticación

Al registrarse o iniciar sesión, la solicitud es gestionada por la instancia de backend aplicando su funcionalidad de registro de actividad.

Esta instancia se conecta directamente con la base de datos, que almacena los datos personales del usuario y su registro de actividad.

Acceso y uso de servicios AI

Una vez autenticado, el usuario accede a la Main service page dentro del frontend. Donde se puede realizar consultas al servicio de AI:

Se genera una petición desde la página principal.

La instancia Engix dirige la petición al backend

Esta es gestionada por la instancia de backend especializada en IA.

Luego es redirigida a una computadora externa al sistema en la nube (hub computer) que procesa la lógica IA y devuelve los resultados al usuario por la misma ruta

Nota sobre Representación

La nube alrededor de componentes como el frontend, backend y base de datos indican que estos corren en una nube (Específicamente de Openstack).

Se omite en el diagrama el paso intermedio por ngnix en cada solicitud a instancias, ya que fue explicado previamente y haría que el diagrama fuera demasiado grande. También se da por hecho que todas las interacciones entre cliente y servicio ocurren a través de su respectiva interfaz visual en la computadora del usuario por lo cual se omite la visualización del proceso con el fin de hacer más legible el diagrama

Arquitectura de la Red

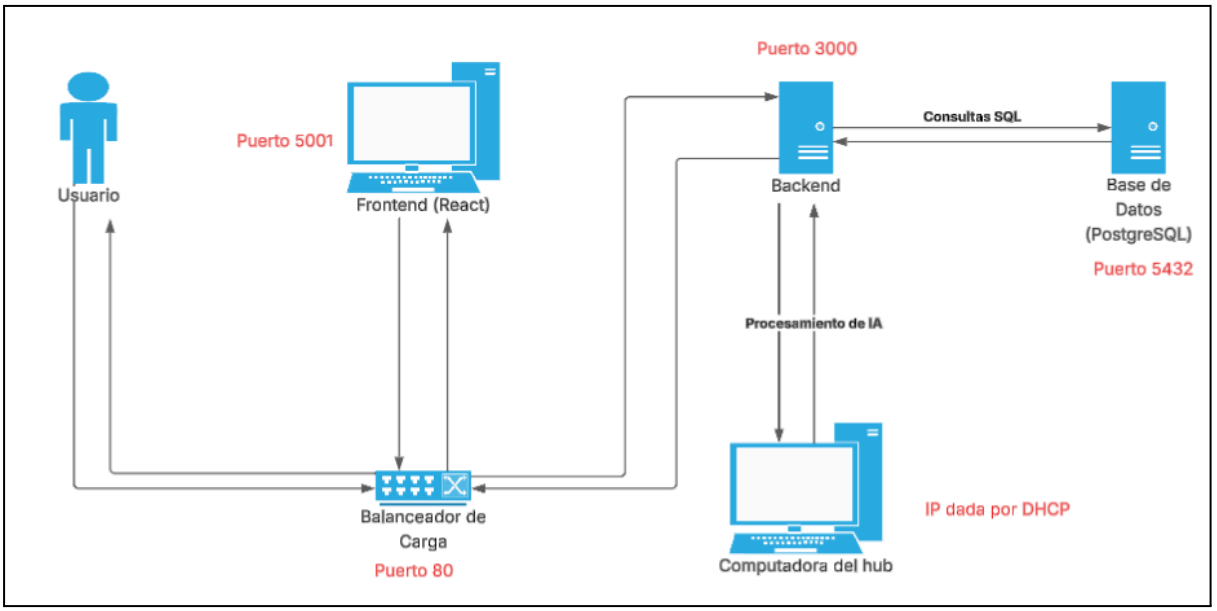


Imagen 3. Creado en: LucidChart.

En esta vista compacta se representa la arquitectura de red utilizada por el sistema, destacando las conexiones entre los distintos componentes y los puertos específicos utilizados por cada uno. También se incluye información sobre la asignación dinámica de IP por DHCP para el equipo encargado del procesamiento de IA.

Se optó por este tipo de representación porque ofrece una visión más operativa y tangible del sistema, tal como sería percibido desde la perspectiva interna de la red, facilitando la comprensión de los flujos de comunicación y el funcionamiento general del entorno.

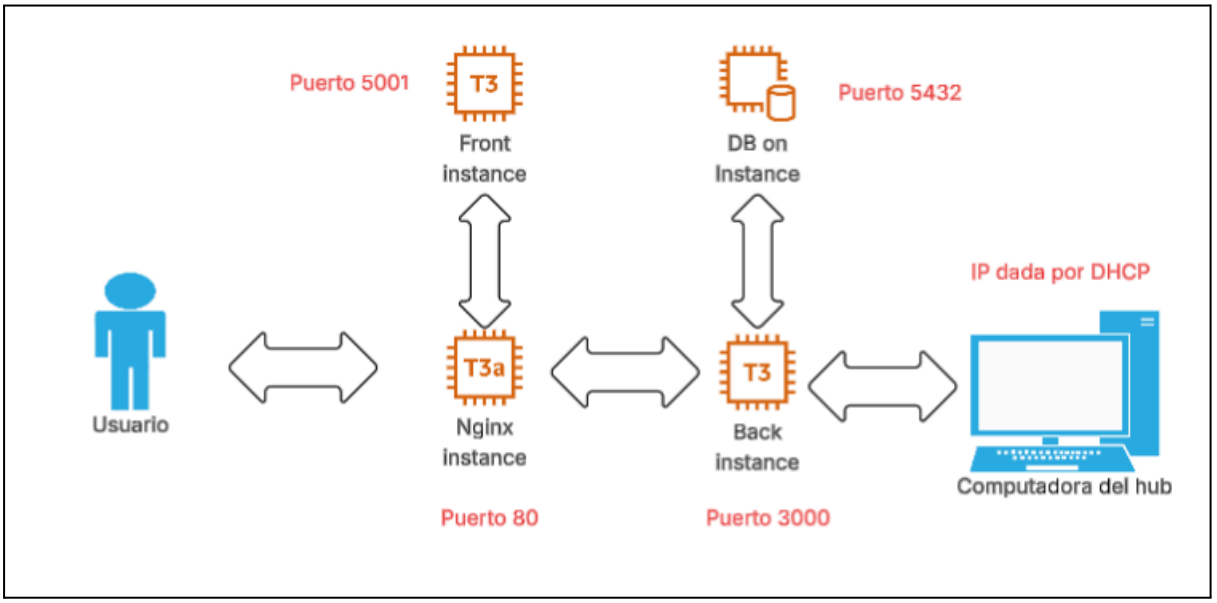


Imagen 4. Creado en: LucidChart.

Esta imagen muestra el despliegue de la arquitectura anterior, indicando en qué instancias dentro de OpenStack se ejecuta cada componente del sistema. Aunque los roles y las conexiones se mantienen, esta vista resulta fundamental para tareas de administración, monitoreo, escalabilidad y mantenimiento de la infraestructura en la nube.

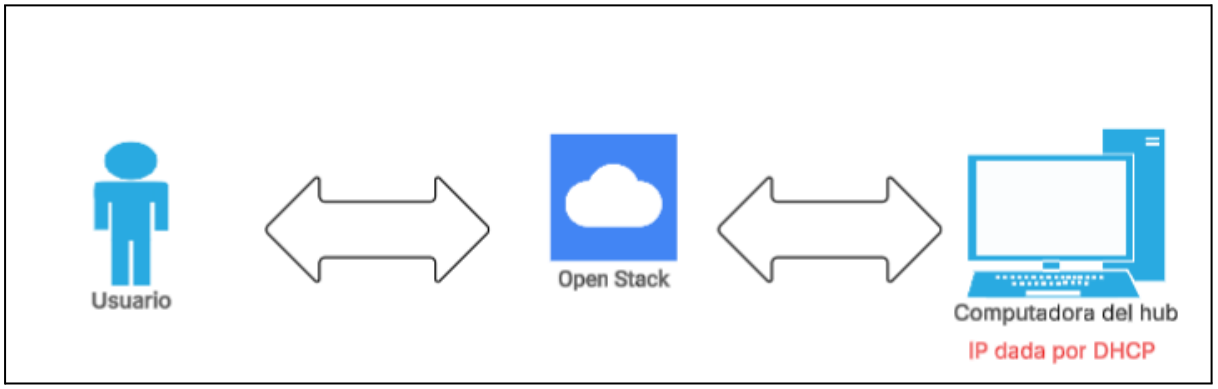


Imagen 5. Creado en: LucidChart.

Finalmente, esta última imagen representa una vista simplificada de la red desde la perspectiva de un observador externo. Aunque se trata del mismo sistema, aquí se abstraen los detalles internos para mostrar únicamente la conexión general entre el usuario, la nube de OpenStack, donde reside la mayor parte de la aplicación y la computadora del hub encargada del procesamiento de IA.

Topología de la Red

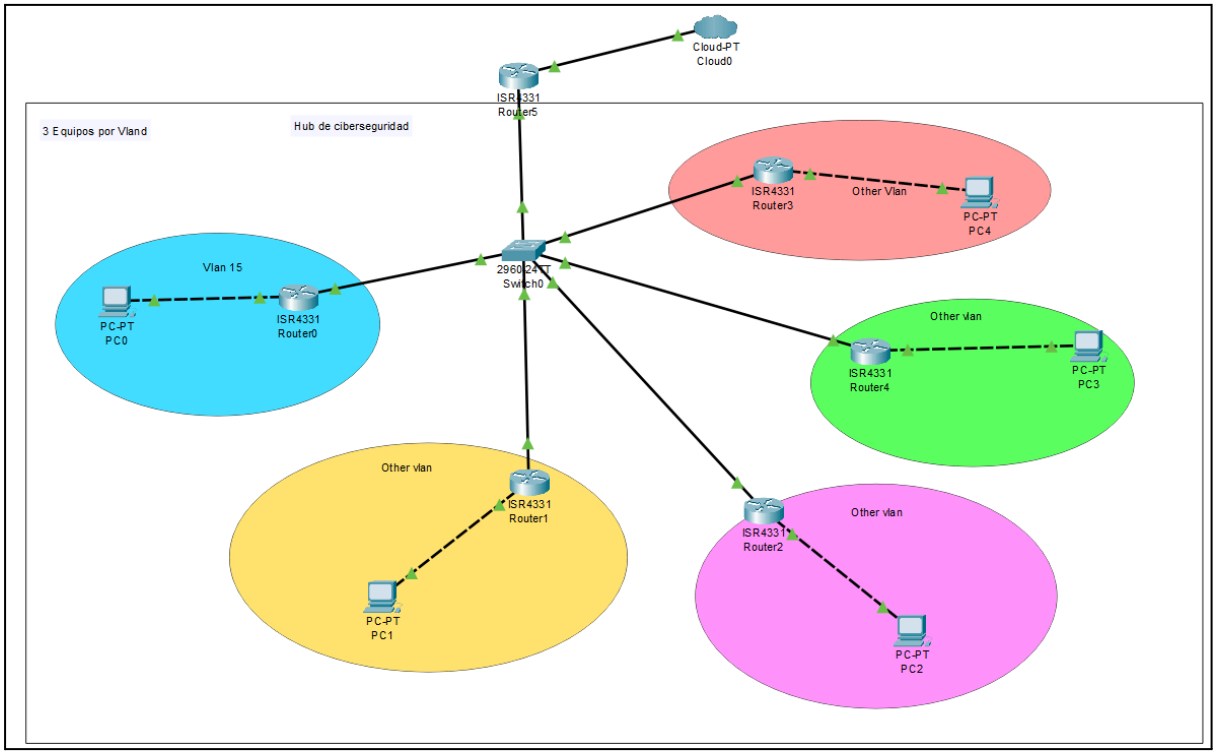


Imagen 6. Creado en: Cisco Packet Tracer.

Esta imagen representa la topología lógica aproximada de la infraestructura de red utilizada durante el desarrollo del proyecto. Se representan las distintas VLAN asignadas a cada equipo, incluyendo la VLAN 15 correspondiente a nuestro grupo, conectadas todas a través de un switch central.

Cada VLAN cuenta con su propio router y un equipo terminal, y están conectadas a un switch que a su vez se enlaza con un router principal (Router5) con acceso hacia una nube.

Esta infraestructura permitió establecer conectividad controlada entre los diferentes entornos de trabajo, así como integrar componentes clave del sistema, como la computadora del hub encargada del procesamiento de IA con el resto de la arquitectura desplegada en OpenStack.

Aunque el entorno formaba parte de un hub compartido, su propósito en este proyecto fue puramente funcional: facilitar pruebas de red, validación de la conectividad entre instancias y simulación de un entorno distribuido realista.

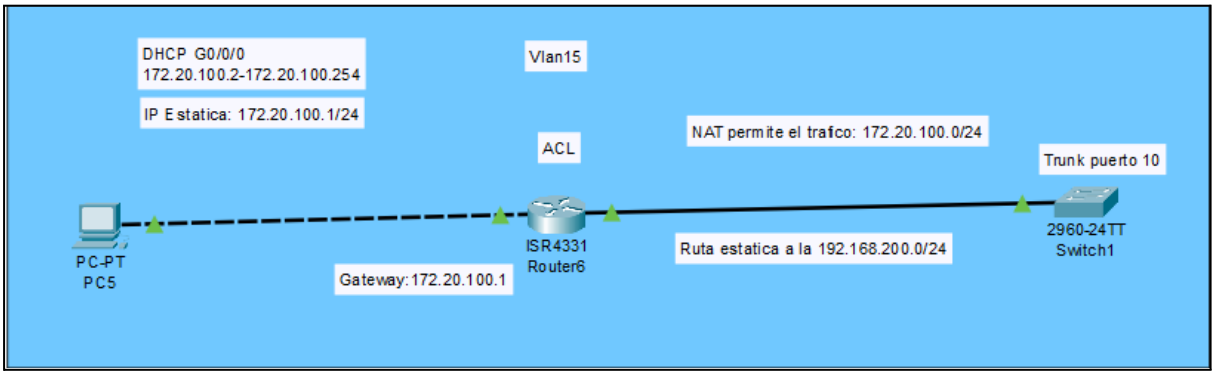


Imagen 7. Creado en: Cisco Packet Tracer.

La imagen muestra la topología de red correspondiente a la VLAN 15, la cual representa el entorno exclusivo configurado por nuestro equipo dentro del proyecto. Esta topología incluye un host (PC5) y un router (Router6) conectados mediante un enlace físico directo, y a su vez enlazados con el resto de la infraestructura a través de un switch.
Nuestro trabajo se centró únicamente en la configuración del router y del host, ya que el resto de los dispositivos y la infraestructura general de red fueron provistos y gestionados externamente.
El host obtiene su dirección IP de forma dinámica a través del servicio DHCP habilitado en la interfaz GigabitEthernet 0/0/0 del router, con un rango de asignación entre 172.20.100.2 y 172.20.100.254. La dirección estática del router en esa misma interfaz es 172.20.100.1/24, actuando como puerta de enlace predeterminada para el host.
El Router6 fue configurado con los siguientes servicios:

- Servidor DHCP, para la asignación automática de IPs dentro de la red local.
- ACL (Lista de Control de Acceso), para filtrar y controlar el tráfico de entrada/salida.

- NAT (Traducción de Direcciones de Red), para permitir la salida de la red local hacia redes externas.

- Ruta estática hacia la red 192.168.200.0/24, con el fin de habilitar la comunicación con redes remotas.

El enlace entre el router y el switch se realiza mediante el puerto 10, el cual se encuentra configurado como trunk para permitir el paso del tráfico de múltiples VLANs, incluida la VLAN 15. Esta configuración aislada en la VLAN 15 permite mantener la segmentación lógica del tráfico y facilita el control sobre los componentes propios del equipo, sin interferir con el resto de la red del proyecto.

Infraestructura Física

Egade piso 5

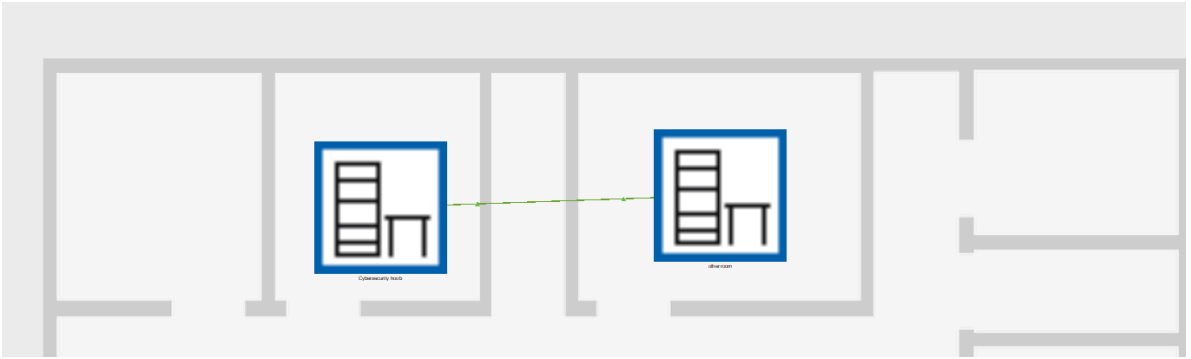


Imagen 8. Creado en: Cisco Packet Tracer.

Cybersecurity HUB

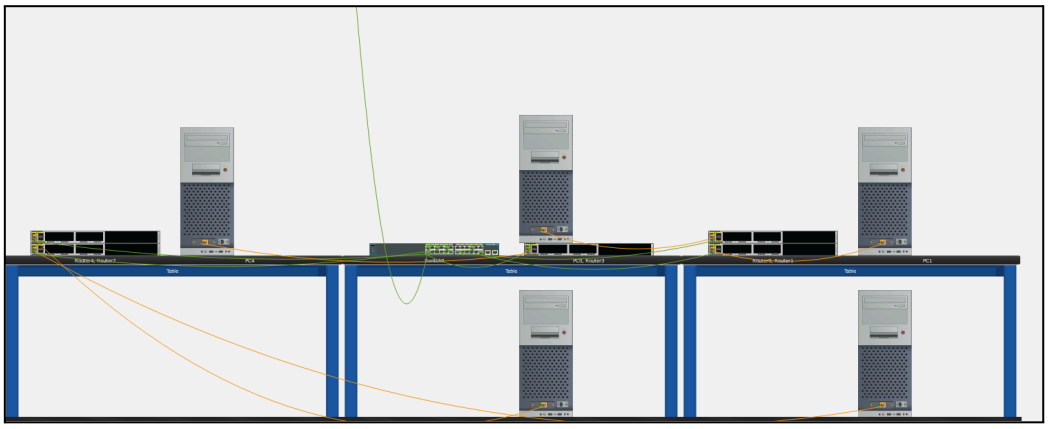


Imagen 9. Creado en: Cisco Packet Tracer.

Imagen 10. Creado en: Cisco Packet Tracer.

Esta imagen corresponde a la vista física del entorno de red, donde se representa gráficamente cómo estarían dispuestos físicamente los dispositivos utilizados en el proyecto. Aunque no era un requerimiento explícito, se decidió incluir esta representación para proporcionar un contexto visual adicional que complemente las topologías lógicas previamente mostradas.

En esta vista se pueden identificar los distintos routers, switches y equipos terminales de cada grupo de trabajo, todos distribuidos en espacios que simulan un entorno real de laboratorio. También se visualiza la infraestructura compartida como el switch central y el router de salida hacia la nube que conecta a todos los equipos. Esta representación es útil para entender la disposición física de la red, facilitar su interpretación espacial y reforzar la idea de segmentación por VLANs dentro de un entorno común.

Frontend

Tecnologías Utilizadas

- ***React:*** El framework principal para construcción de toda la interfaz de usuario que incluye los componentes funcionales. Se utilizó CSS para estilizar la página.
- ***React Router DOM:*** Biblioteca para navegación y enrutamiento desde el lado cliente.
- ***Testing Libraries:*** Jest DOM, React Testing Library y User Event para realizar las pruebas unitarias
- ***Dotenv:*** Gestión de variables de entorno para configuración
- ***Web Vitals:*** Métricas de rendimiento web para optimización

La arquitectura del frontend se organizó mediante componentes reutilizables agrupados por vistas o funciones específicas (como Login o ChatBot), estructurados en carpetas para facilitar la escalabilidad y el mantenimiento. La gestión de estados se implementó con useState para manejar estados locales dentro de cada componente y useContext para compartir información global como la autenticación, permitiendo un flujo de datos claro y eficiente entre componentes.

Backend

Tecnologías Utilizadas

- ***Node.js:*** con Express 5.1.0: Framework principal del servidor web
- ***Prisma 6.8.2:*** ORM para gestión de base de datos y migrations
- ***Babel:*** Transpilador para uso de ES6 modules y sintaxis moderna
- ***JWT (jsonwebtoken 9.0.2):*** Manejo de tokens para autenticación
- ***Bcrypt 6.0.0:*** Cifrado de contraseñas con hash seguro
- ***Axios 1.9.0:*** Cliente HTTP para comunicación con servicios externos
- ***Jest 29.7.0:*** Framework de testing para pruebas unitarias

Arquitectura Interna

1. **Capa de presentación:** Controladores *auth.controller.js, conversation.controller.js* manejan requests HTTP.
2. **Capa de lógica:** Dentro de los controladores se implementan las validaciones, reglas de negocio y transformaciones específicas del proyecto, como el flujo de autenticación y el manejo de peticiones al modelo de IA.
3. **Capa de acceso a datos:** Prisma ORM para interacción con PostgreSQL.

Configuraciones de Seguridad

- ***Express Rate Limit:*** Máximo permite 100 peticiones por IP cada 15 minutos, esto para prevenir ataques DDoS y abuso.
- ***Helmet:*** Configuración de headers de seguridad (CSP, HSTS, frameguard) para prevenir XSS y clickjacking
- ***CORS:*** Configurado para permitir solo orígenes específicos (localhost:5001 en dev, frontend.misitio.com en prod)
- ***Content Security Policy:*** Permite solo scripts del mismo origen y Google APIs, bloquea objetos externos
- ***Morgan:*** Logging de todas las peticiones HTTP para auditoría y debugging

API y Comunicación

Endpoints y Comunicación

Endpoint	Método	Funcionalidad	Payload	Respuesta	Estado
/auth/register	POST	Registrarse	{"campus", "name", "lastName", "email", "password"}	{state, message, user, token}	201/400/409
/auth/login	POST	Iniciar Sesión	{"email", "password"}	{state, message, user, token}	200/400/401
/auth/logout	GET	Cerrar sesión	-	Redirección al login	200
/conversation	POST	Consulta al chatbot	{"prompt"}	{response}	200/400/500

Manejo de Errores, Formato y Validación

Código	Descripción	Validación/Formato	Mensaje de respuesta
200	Login exitoso	Email @tec.mx válido, contraseña correcta	"Login successful"
201	Usuario creado	Email @tec.mx, contraseña 8+ caracteres	"User created successfully"
400	Datos inválidos	Email/password faltantes, prompt vacío	"Email and password are required" / "Prompt is required"
401	No autorizado	Credenciales incorrectas, usuario no encontrado	"Invalid email or password" / "User Not Found"
409	Conflicto	Usuario ya registrado	"User already exists"
500	Error interno	Fallo en base de datos, modelo IA, servidor	"Internal server error" / "Fail"

Base de Datos

Resumen del Diseño y justificación

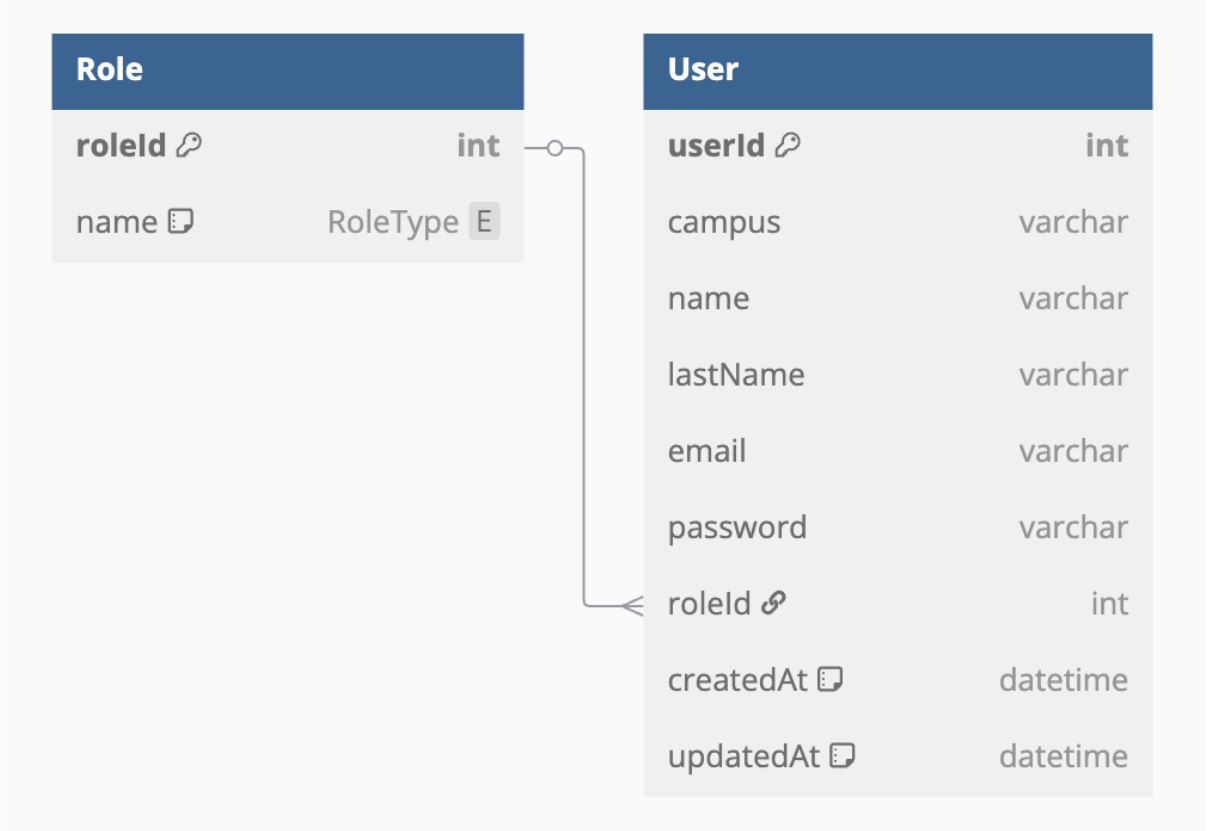


Imagen #11. Recuperado de: https://dbdiagram.io/.

El esquema de base de datos representado consta de dos tablas principales: User y Role. Este diseño está orientado a gestionar información relacionada con usuarios y los distintos niveles de acceso que estos pueden tener dentro de un sistema. Es ideal para aplicaciones que requieren control de autenticación y autorización, como plataformas académicas, sistemas administrativos, o entornos corporativos donde diferentes perfiles de usuario deben acceder a funcionalidades específicas.

La tabla User almacena la información esencial de cada persona registrada en el sistema. Cada usuario cuenta con un identificador único (userId) que funciona como clave primaria. Adicionalmente, se registran atributos como el campus al que pertenece, su nombre y apellido, dirección de correo electrónico, y la contraseña que se utilizará para autenticarse en el sistema. Estos datos permiten identificar y validar de manera individual a cada usuario. A

su vez, se incluye un campo llamado roleId, que actúa como clave foránea hacia la tabla Role, indicando qué tipo de permisos o perfil tiene asignado ese usuario. Finalmente, los campos createdAt y updatedAt registran las fechas de creación y última modificación del registro, lo que permite un seguimiento cronológico útil en tareas de auditoría o mantenimiento.

Por su parte, la tabla Role tiene la función de clasificar y describir los distintos tipos de roles que existen dentro del sistema. Contiene únicamente dos campos: roleId, que es la clave primaria y se utiliza como referencia en la tabla User, y name, que describe el tipo de rol, como por ejemplo “Administrador”, “Estudiante”, “Docente”, entre otros. Este campo puede estar vinculado a un tipo de enumeración (RoleType), lo cual estandariza el uso de nombres y evita errores semánticos.

La relación entre ambas tablas es de tipo uno a muchos: un mismo rol puede estar asignado a múltiples usuarios, pero cada usuario solo puede tener un único rol asociado. Esta relación está bien definida a través de la clave foránea roleId en la tabla User, lo que permite un control efectivo y organizado del acceso al sistema con base en los permisos asociados a cada rol.

Desde el punto de vista de la normalización, este modelo cumple con las reglas de la tercera forma normal (3FN). Cada campo contiene datos atómicos, no existen dependencias transitivas dentro de una misma tabla, y se evita la duplicidad de datos. Además, al separar los roles en una tabla específica, se facilita la administración y escalabilidad del sistema, permitiendo la creación de nuevos tipos de usuario sin afectar la estructura base.

El diseño explicado anteriormente hace que la base de datos sea sólida, flexible y extensible, ofreciendo una solución eficiente para manejar usuarios y roles en cualquier sistema que requiera autenticación, control de acceso y trazabilidad de usuarios. Su implementación facilita el mantenimiento del sistema y la integración de futuras funcionalidades sin comprometer la integridad de los datos.

Seguridad

Autenticación JSON Web Tokens y Cifrado

- Tokens generados automáticamente en login/registro con userId y expiración de 1 hora
- Firmados con clave secreta fija
- Validación automática de tokens antes de acceder a endpoints
- Bcrypt con factor de salt 10, nunca se almacena texto plano en base de datos
- De la misma manera, contamos con certificados de seguridad en la página y https

Validaciones y controles de acceso

- Email obligatorio formato @tec.mx con validation
- Sanitización automática de entradas JSON mediante Express.json()
- Sistema de rate limiting implementado con máximo 100 peticiones por IP cada 15 minutos
- CORS configurado dinámicamente permitiendo localhost:5001 en desarrollo y frontend.misitio.com en producción

Históricos y Manejo de Sesiones

- Morgan registra automáticamente todas las peticiones HTTP en modo desarrollo
- El sistema realiza tracking de intentos de login exitosos y fallidos
- La auditoría detallada de acciones de usuario está pendiente de expandir
- Los tokens JWT tienen expiración automática de 1 hora y se almacenan en localStorage
- El logout elimina el token del localStorage aunque la invalidación en el servidor está pendiente
- Las preferencias de tema se persisten entre visitas a la aplicación
- La continuidad de sesión se mantiene mientras el token sea válido

Red

Infraestructura en Nube Privada con OpenStack

OpenStack es una plataforma de cloud computing de código abierto que permite crear y gestionar nubes privadas. Funciona dentro de la propia infraestructura, ofreciendo control total sobre recursos, datos y configuraciones. Implementamos TecGPT utilizando 4 instancias dedicadas en nuestra nube privada OpenStack:

	1. Frontend (React)	**2. Backend (Node.js)**	**3. Base de Datos (PostgreSQL)**	**4. Balanceador (NGINX)**
Propósito	Servir la aplicación web React	API REST y lógica de negocio	Almacenamiento persistente de usuarios y logs	Load balancer y proxy reverso
Configuración Servicios	Servidor web con archivos estáticos optimizados	Autenticación JWT, integración con IA, validaciones	PostgreSQL optimizada con backups automáticos	Distribución de tráfico, SSL/TLS, caching, rate limiting
Puerto	5001	3000	5432	80/443

Plan de Conexión a la Nube

Se crearon estas 4 instancias en OpenStack con sus respectivos grupos de seguridad para establecer una arquitectura de aplicación distribuida que incluye componentes de frontend, backend, base de datos y balanceador de carga.

Configuración de Instancias

Sistema Operativo: Debian (como imagen base)

Flavor: Tiny (configuración de recursos mínima)

Región: OpenStack

Autenticación: Llave TecGPT (keypair para acceso SSH)

Asignación de IPs en la Nube Privada

Las direcciones IP pertenecen a la subred 172.20.100.0/24, configurada dentro de la nube privada gestionada con OpenStack. Cada instancia cumple un rol específico dentro de la arquitectura modular de TecGPT, lo que permite separar responsabilidades, facilitar el mantenimiento y mejorar la seguridad. Frontend y Backend están desplegados en servidores independientes para optimizar el rendimiento y aislar fallos. La base de datos está en su propia instancia para mantener la persistencia de datos y asegurar backups independientes del resto del sistema.

Configuración de Dispositivos de Red

Configuración de Router

```
enable
configure terminal
interface GigabitEthernet0/0/0
ip address dhcp
ip nat outside
no shutdown
exit
interface GigabitEthernet0/0/1
no ip address
no shutdown
exit
interface GigabitEthernet0/0/1.15
encapsulation dot1Q 15
ip address 172.20.100.1 255.255.255.0
ip nat inside
no shutdown
exit
```

```
no ip dhcp pool VLAN15_POOL
no ip dhcp excluded-address 192.168.15.1
ip dhcp excluded-address 172.20.100.1
ip dhcp pool VLAN15_POOL
network 172.20.100.0 255.255.255.0
default-router 172.20.100.1
dns-server 8.8.8.8
exit
ip route 192.168.200.0 255.255.255.0 172.20.100.254
no access-list 1
no ip nat inside source list 1 interface GigabitEthernet0/0/0 overload
access-list 1 permit 172.20.100.0 0.0.0.255
ip nat inside source list 1 interface GigabitEthernet0/0/0 overload
end
copy running-config startup-config
```

Configuración de Grupos de Seguridad

1. **NGNIXGPT Security Group**
   Propósito: Servidor web/proxy reverso
   Reglas de Seguridad:

Tipo	Protocolo	Puerto(s)	Origen	Descripción
Egress	IPv4	Any	0.0.0.0/0	Permite todo tráfico saliente
Egress	IPv6	Any	::/0	Permite todo tráfico saliente
Ingress	ICMP	Any	0.0.0.0/0	Permite ping desde cualquier origen
Ingress	TCP	Any	0.0.0.0/0	Permite todo tráfico TCP entrante
Ingress	TCP	22 (SSH)	0.0.0.0/0	Acceso remoto por SSH
Ingress	TCP	80 (HTTP)	0.0.0.0/0	Acceso web HTTP

2. **FrontendGPT Security Group**
   Propósito: Servidor de aplicación frontend
   Reglas de Seguridad:

Tipo	Protocolo	Puerto(s)	Origen	Descripción
Egress	IPv4	Any	0.0.0.0/0	Permite todo tráfico saliente
Egress	IPv6	Any	::/0	Permite todo tráfico saliente
Ingress	ICMP	Any	0.0.0.0/0	Permite ping desde cualquier origen
Ingress	TCP	22 (SSH)	0.0.0.0/0	Acceso remoto por SSH
Ingress	TCP	80 (HTTP)	0.0.0.0/0	Acceso web HTTP
Ingress	TCP	443 (HTTPS)	0.0.0.0/0	Acceso web HTTPS
Ingress	TCP	5001	0.0.0.0/0	Puerto personalizado para el frontend

3. **DatabaseGPT Security Group**

Propósito: Servidor de base de datos
Reglas de Seguridad:

Tipo	Protocolo	Puerto(s)	Origen	Descripción
Egress	IPv4	Any	0.0.0.0/0	Permite todo tráfico saliente
Egress	IPv6	Any	::/0	Permite todo tráfico saliente
Ingress	ICMP	Any	0.0.0.0/0	Permite ping desde cualquier origen
Ingress	TCP	22 (SSH)	0.0.0.0/0	Acceso remoto por SSH
Ingress	TCP	80 (HTTP)	0.0.0.0/0	Acceso web HTTP
Ingress	TCP	443 (HTTPS)	0.0.0.0/0	Acceso web HTTPS
Ingress	TCP	5432	0.0.0.0/0	Puerto PostgreSQL para consultas de la BD

4. **BackendGPT Security Group**
   Propósito: Servidor de aplicación backend
   Reglas de Seguridad:

Tipo	Protocolo	Puerto(s)	Origen	Descripción
Egress	IPv4	Any	0.0.0.0/0	Permite todo tráfico saliente
Egress	IPv6	Any	::/0	Permite todo tráfico saliente
Ingress	ICMP	Any	0.0.0.0/0	Permite ping desde cualquier origen
Ingress	TCP	Any	0.0.0.0/0	Permite todo tráfico TCP entrante
Ingress	TCP	22 (SSH)	0.0.0.0/0	Acceso remoto por SSH
Ingress	TCP	3000	0.0.0.0/0	Puerto personalizado para el backend
Ingress	TCP	5432	0.0.0.0/0	Acceso a base de datos PostgreSQL

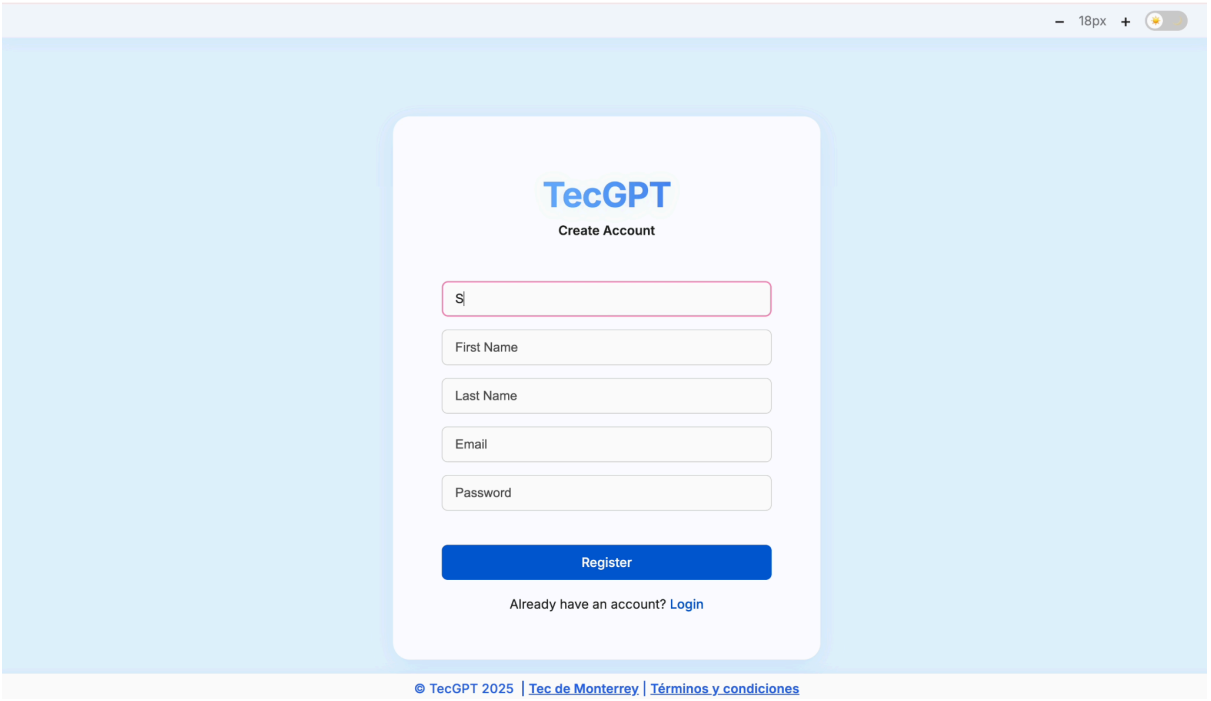
Flujo de Datos

Usuario → Nginx (Load Balancer) → Frontend (React) → Backend (Node.js) →PostgreSQL
↓
Modelo IA (Servicio Externo)

Configuración de Llaves SSH

La configuración de autenticación SSH para todas las instancias del proyecto se realizó mediante una llave llamada TecGPT Key Pair. Esta llave es un par de claves del tipo RSA, utilizada para habilitar el acceso seguro a través del protocolo SSH. La clave fue generada directamente en la plataforma OpenStack, aunque también puede ser importada desde otro origen en función de las necesidades del proyecto. Su propósito principal es garantizar la autenticación segura y sin contraseñas en el acceso remoto a las instancias.

UI/UX



Diseño Responsivo

TecGPT implementa un diseño responsivo que se adapta a dispositivos móviles, tablets y escritorio, asegurando una experiencia consistente en todas las plataformas. La aplicación incluye modo claro y oscuro para reducir fatiga visual, con efectos hover en botones e imágenes que proporcionan retroalimentación inmediata al usuario. El sistema es accesible ya que utilizamos un contraste adecuado de colores y tamaños de fuente configurables, garantizando usabilidad para usuarios con diferentes capacidades y necesidades visuales.

GitHub del Repositorio TecGPT

Link al repositorio: *https://github.com/MiguelCabreraVictoria/TecGPT*

Pruebas Realizadas

Casos de Uso - Pruebas

Tipo de Prueba	Caso de Uso Probado	Caso de Prueba	Resultado Esperado	Estado
Frontend	CU-01: Registro nuevo usuario	Validación campos requeridos en formulario	Error si faltan campos obligatorios	Exitoso
Frontend	CU-01: Registro nuevo usuario	Formato email @tec.mx	Rechazo de emails con formato incorrecto	Exitoso
Frontend	CU-02: Inicio de sesión	Redirección automática con token válido	Usuario redirigido a /chat tras login exitoso	Exitoso
Frontend	CU-04: Cambiar tema visual	Persistencia de preferencias	Tema seleccionado se mantiene entre páginas	Exitoso
API	CU-05: Autenticar credenciales	Login con credenciales válidas	Respuesta 200 con token y datos usuario	Exitoso
API	CU-05: Autenticar credenciales	Login con credenciales inválidas	Respuesta 401 Unauthorized con mensaje error	Exitoso
API	CU-06: Crear nuevo usuario	Registro con email no @tec.mx	Respuesta 400 Bad Request por formato inválido	Exitoso
API	CU-06: Crear nuevo usuario	Registro con contraseña menor a 8 caracteres	Respuesta 400 Bad Request por política contraseñas	Exitoso
API	CU-07: Procesar consulta IA	Consulta con token válido	Respuesta 200 con datos del modelo IA	Exitoso

API	CU-07: Procesar consulta IA	Consulta sin token de autorización	Respuesta 401 Unauthorized	Exitoso
Base de Datos	CU-08: Gestionar usuarios	Creación usuario con datos únicos	Usuario almacenado correctamente en tabla users	Exitoso
Base de Datos	CU-09: Registrar actividades	Logging de intento de login	Registro insertado en tabla logs con timestamp	Exitoso
Integración	CU-03: Consulta administrativa	Flujo completo registro, login y chat	Usuario puede completar flujo sin errores	Exitoso
Seguridad	Rate Limiting	Más de 100 peticiones en 15 minutos	Respuesta 429 Too Many Requests	Exitoso
Seguridad	CU-06: Crear nuevo usuario	Intento de registro con email duplicado	Respuesta 409 Conflict usuario ya existe	Exitoso

Las pruebas realizadas en TecGPT han sido exitosas. Las pruebas de frontend con Jest y validaciones manuales de base de datos confirmaron que todos los componentes críticos funcionan apropiadamente, incluyendo autenticación, registro de usuarios, consultas al chatbot y medidas de seguridad.

Manual de Usuario
Link al Manual de Usuario:
https://drive.google.com/drive/folders/1h2x0nk7fWdI86Hmmn-lvDgYxDBn8tGCT?usp=sharing

Conclusiones

El desarrollo de TecGPT representó un desafío técnico exitoso que nos permitió crear una solución real para las consultas administrativas frecuentes en el Tecnológico de Monterrey Campus. Logramos implementar un modelo de inteligencia artificial personalizado entrenado con 700 preguntas específicas del campus, desarrollar una aplicación web completa usando React y Node.js/Express, y establecer un sistema de autenticación seguro que incluye características de accesibilidad y medidas de seguridad robustas. Aunque reconocemos ciertas limitaciones como la operación únicamente en inglés debido a las complejidades del procesamiento en español, consideramos que estas representan oportunidades de crecimiento más que obstáculos. TecGPT presenta múltiples oportunidades de expansión, incluyendo la posibilidad de ampliar el modelo para soportar español, integrar con bases de datos académicas institucionales para información en tiempo real, utilizar la configuración de Docker ya desarrollada para escalamiento automático cuando se requiera mayor capacidad, y la potencial incorporación de funcionalidades avanzadas como notificaciones push e integración con sistemas de calendarios académicos que podrían transformar significativamente la experiencia estudiantil. El éxito del proyecto se evidencia no solo en su funcionalidad técnica, sino en la capacidad del equipo para desarrollar una solución completa y operativa en pocos días.