Tecnológico de Monterrey

Planeación de sistemas de software Gpo. 103

Especificación de diseño y arquitectura

Docente:

Juan Carlos Lavariega Jarquín

Equipo “The Bytles”:

Ana Karina Aramoni Ruíz - A07192068

Yuting Lin - A00835917

René Miguel Macías Olivar - A00836714

Eugenio Andrés Mejía Fanjón - A01412143

Pedro Enrique Gómez Palafox - A01027841

Rodrigo Garza de la Rosa - A01383556

Fecha: 6/6/2025

ÍNDICE

1. Descripción General del Proyecto y Objetivos (Quality Goals)

PathExplorer es una plataforma de gestión del desarrollo profesional diseñada para centralizar y optimizar la toma de decisiones estratégicas sobre talento en Accenture. El sistema consolida datos clave del ciclo de vida del empleado (como historial, metas, habilidades, y asignaciones) y habilita funcionalidades de análisis, recomendación y visualización que respaldan la asignación eficiente de roles.

Objetivos de calidad definidos:

- ***Seguridad***: Implementación de control de acceso basado en roles (RBAC) y validación de seguridad mediante pruebas de penetración.
- ***Disponibilidad y rendimiento:*** Tiempo de respuesta inferior a 1.5 segundos en al menos el 60% de las transacciones críticas.
- ***Mantenibilidad:*** Modularidad arquitectónica y documentación clara que facilite el desarrollo evolutivo y la corrección de errores.
- ***Escalabilidad:*** Capacidad de atender un crecimiento progresivo de usuarios y volumen de datos sin degradación significativa en el rendimiento.
- ***Compatibilidad:*** Soporte para navegadores modernos (Chrome, Firefox, Edge, Safari) y plataformas Windows/macOS.

2. Contexto, Restricciones y Alcance

Accenture enfrenta una fragmentación significativa en la información sobre talento, lo cual limita la visibilidad, la planificación y la asignación óptima de roles. PathExplorer responde a este desafío como una solución interna unificada (sin dependencias externas en su versión inicial), diseñada para facilitar el análisis de trayectorias profesionales y procesos de staffing.

Restricciones técnicas y operativas:

- ***Stack tecnológico obligatorio:*** React, JavaScript, Supabase (PostgreSQL).
- ***Compatibilidad multiplataforma:*** Windows/macOS y navegadores previamente mencionados.
- ***Infraestructura controlada:*** Máquina virtual compartida sobre Linux, limitada a ventanas de operación definidas.
- ***Privacidad de la retroalimentación:*** Visibilidad de comentarios y feedback restringida a usuarios con privilegios de nivel 2 (managers).
- ***Capacidades de IA:*** El uso de servicios de inteligencia artificial para recomendaciones es opcional pero altamente recomendado

Alcance funcional del sistema:

- Gestión integral de perfiles de usuario (CVs, biografía, metas, habilidades, certificaciones).

- Administración de proyectos y roles asignados.
- Motor de comparación semántica entre perfil y requerimientos de rol.
- Generación de recomendaciones personalizadas a través de modelos de lenguaje.
- Visualización de trayectorias profesionales, métricas clave (KPIs) y reportes de asignación.

3. Estrategia de Solución

Tecnologías clave:

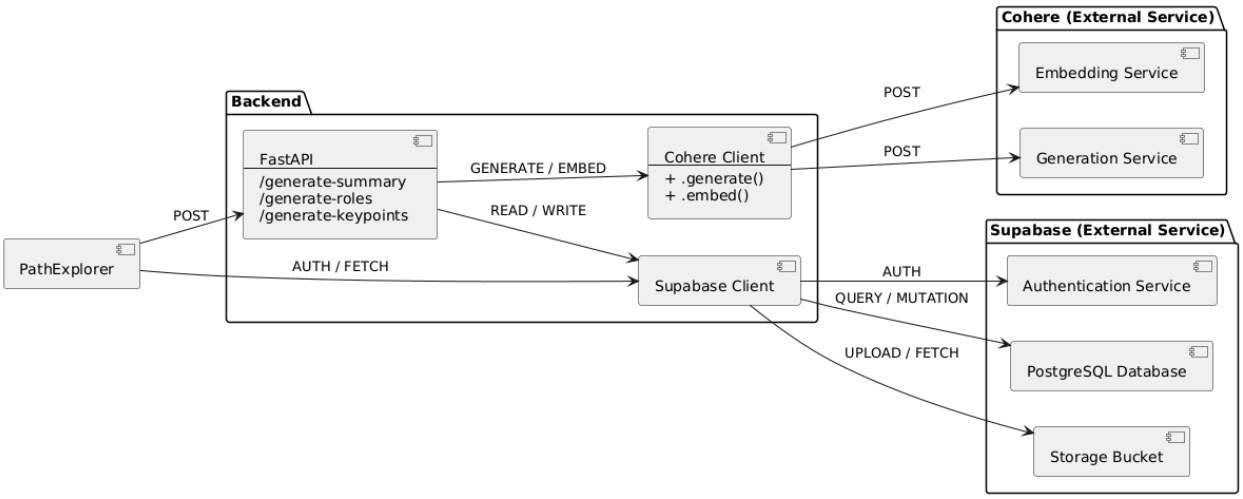
- ***Frontend:*** React + Vite (composición modular, experiencia dinámica)
- ***Backend:*** FastAPI (coordinador de lógica e integración con IA)
- ***Base de datos y autenticación:*** Supabase (PostgreSQL + Auth)
- ***Almacenamiento:*** Supabase Storage (archivos .PDF, .PNG)
- ***Estilo arquitectónico:*** La solución se implementa bajo una arquitectura Cliente–Servidor con enfoque orientado a microservicios. El cliente (frontend) está desarrollado con un enfoque modular construido en React, donde cada vista se compone de componentes reutilizables y reactivos que gestionan tanto la lógica de presentación como la interacción del usuario. En el backend, se integran dos servicios diferenciados:
    - Supabase, adoptado como Backend-as-a-Service (BaaS), proporciona: autenticación y autorización con control granular, persistencia estructurada sobre PostgreSQL; y almacenamiento de documentos y recursos mediante buckets.
    - *FastAPI* que expone una serie de endpoints REST (/generate-summary, /generate-roles, /generate-keypoints) y actúa como orquestador de lógica compleja y punto de integración con Cohere, una API externa de inteligencia artificial. Este backend también se comunica con Supabase para realizar operaciones adicionales de lectura/escritura.

  A nivel de IA, la solución incorpora llamadas asincrónicas a Cohere API, para realizar generación automática de texto y embeddings semánticos. Esto habilita mecanismos de recomendación personalizados y análisis contextualizados, siguiendo un enfoque event-driven compatible con la naturaleza reactiva del frontend.

Tácticas de calidad aplicadas:

- ***Pruebas:*** Combinación de pruebas manuales y automatizadas (unitarias, funcionales).
- ***Métricas y mantenimiento:*** Revisión de código, métricas de cobertura y tiempo de corrección de bugs.
- ***Escalabilidad horizontal:*** Uso de base de datos cloud y desacoplamiento de servicios.
- ***Metodología ágil:*** Ciclos de desarrollo en Sprints de 2 semanas bajo marco SCRUM, gestionados mediante Jira.

4. Diagrama General



5. Descomposición de Subcomponentes

- Frontend (React)

  El cliente web fue desarrollado con React + Vite bajo un patrón component-based, utilizando hooks y composición modular para mantener cohesión y reusabilidad. Las vistas principales incluyen:
  - ***Login & Registro:*** Validación de identidad y creación de usuarios mediante Supabase Auth.
  - ***Perfil de Usuario:*** Visualización y edición de datos personales, CV, skills, metas, cursos y certificaciones.
  - ***Gestión de Proyectos:*** Creación y visualización de proyectos asignados o propuestos; y cálculo de compatibilidad semántica con roles activos.
  - ***Vista de Empleados:*** Dashboard administrativo con acceso segmentado por rol.
  - ***Historial Profesional:*** Seguimiento de asignaciones pasadas por usuario.
  - ***Crecimiento:*** Interfaz para visualizar recomendaciones generadas por IA de actividades, cursos y certificaciones.

- Backend
  - Supabase (BaaS)
    - ***Base de Datos:*** PostgreSQL como motor principal, con 13 tablas normalizadas que representan entidades clave como User, Project, Role, Certificates, Skills, Goals, User_History, entre otras.
    - ***Triggers y RPCs:*** Se utilizan funciones remotas y triggers SQL para:
      - Cálculo de tiempo en banca.
      - Comparación semántica y asignación de compatibilidades.

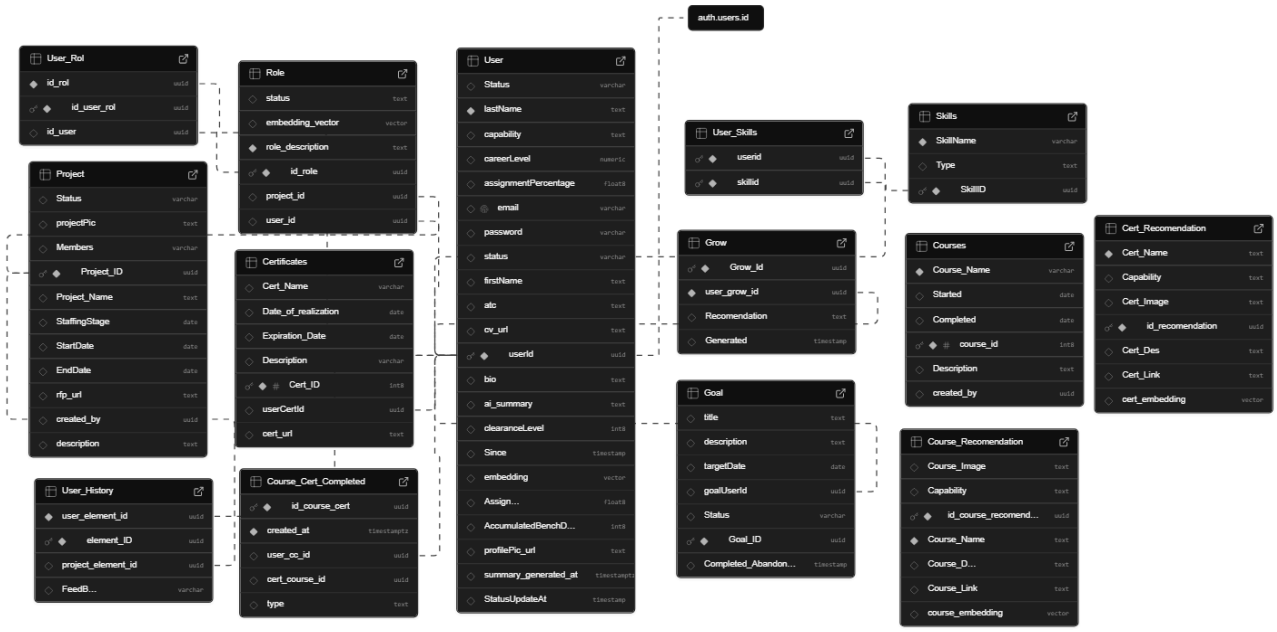
            - Recomendación de cursos y certificaciones mediante embeddings.
    - FastAPI (Service Layer / Orquestador IA)
        - Exposición de endpoints RESTful:
            - POST /generate-summary
            - POST /generate-roles
            - POST /generate-keypoints
        - Encapsula lógica de negocio compleja y maneja llamadas asincrónicas a Cohere API, actuando como adaptador de servicios de IA.
        - También interactúa con Supabase vía su SDK o APIs REST para actualizar campos como ai_summary, embedding, o Grow.

- Infraestructura y Despliegue
    - ***Almacenamiento de archivos:*** Supabase Storage organiza los archivos del sistema bajo las siguientes rutas:
        - ***cv/:*** currículums de usuarios.
        - ***rfp/:*** documentos de especificación de proyecto.
        - ***profilePics/:*** fotos de perfil.
    - Frontend alojado en Vercel, con CI/CD integrado desde GitHub.
    - FastAPI desplegado en una VM Linux compartida, con llamadas externas seguras a Cohere y acceso autenticado a Supabase.

6. Vista lógica

La vista lógica del sistema define las entidades de negocio, sus relaciones y su mapeo a nivel relacional, estableciendo una base sólida para validaciones, lógica de negocio y persistencia.

- **Diagrama UML:**



- **Mapeo a Relacional:**

  User (***user_id***, first_name, last_name, email, password, capability, career_level, assignment_percentage, cv_url, bio, ai_summary, clearance_level, since, embedding, assign_p, accumulated_bench_days, profile_pic_url, summary_generated_at, status, status_updated_at, atc)
  FOREIGN KEY (user_id) REFERENCES auth.users(id)

  User_History (***element_id***, user_id, project_id, feedback)
  FOREIGN KEY (user_id) REFERENCES User(user_id),
  FOREIGN KEY (project_id) REFERENCES Project(project_id)

  User_Rol (***id_user_rol***, id_user, id_rol)
  FOREIGN KEY (id_user) REFERENCES User(user_id),
  FOREIGN KEY (id_rol) REFERENCES Role(id_role)

  User_Skills (***user_id, skill_id***)
  FOREIGN KEY (user_id) REFERENCES User(user_id),
  FOREIGN KEY (skill_id) REFERENCES Skills(skill_id)

  Skills (***skill_id***, skill_name, type)

Project (***project_id***, project_name, description, status, members, staffing_stage, start_date, end_date, rfp_url, project_pic, created_by)
FOREIGN KEY (created_by) REFERENCES User(user_id)

Role (***id_role***, status, role_description, embedding_vector, project_id, user_id)
FOREIGN KEY (project_id) REFERENCES Project(project_id),
FOREIGN KEY (user_id) REFERENCES User(user_id)

Goal (***goal_id***, title, description, target_date, status, completed_abandoned_at, user_id)
FOREIGN KEY (user_id) REFERENCES User(user_id)

Grow (***grow_id***, user_id, recommendation, generated)
FOREIGN KEY (user_id) REFERENCES User(user_id)

Certificates (***cert_id***, cert_name, description, date_of_realization, expiration_date, cert_url, user_id)
FOREIGN KEY (user_id) REFERENCES User(user_id)

Cert_Recomendation (***id_recommendation***, cert_name, capability, cert_image, cert_des, cert_link, cert_embedding)

Courses (***course_id***, course_name, description, started, completed, created_by)
FOREIGN KEY (created_by) REFERENCES User(user_id)

Course_Recomendation (***id_course_recommendation***, course_name, capability, course_des, course_image, course_link, course_embedding)

Course_Cert_Completed (***id_course_cert***, cert_course_id, user_cc_id, type, created_at)
FOREIGN KEY (user_cc_id) REFERENCES User(user_id)

- **Flujo del Usuario:**

El siguiente flujo describe las principales interacciones del usuario dentro de la plataforma PathExplorer, desde el inicio de sesión hasta la asignación de proyectos y visualización de recomendaciones.

1. ***Inicio de sesión y autenticación:*** El usuario accede a la plataforma a través de un flujo de login basado en Supabase Auth. Se valida su identidad y se determina su nivel de acceso (empleado, manager, etc.).

2. ***Registro y configuración de perfil:*** Al ingresar por primera vez, el usuario completa su perfil profesional mediante un formulario que permite:
    a. Carga de CV en formato .PDF
    b. Redacción de biografía
    c. Foto de perfil en formato .PNG
    d. Selección de habilidades técnicas y blandas
    e. Definición de metas profesionales
    f. Registro de cursos y certificaciones previas
    g. Registro de nuevos proyectos (Manager)

3. ***Registro de nuevos proyectos (Manager):*** Los usuarios con rol de manager pueden crear proyectos ingresando:
    a. Título y descripción
    b. Fechas de reclutamiento, inicio y finalización
    c. Documento RFP en formato .PDF
    d. Imagen del proyecto (opcional)
4. ***Generación automática de roles (IA):*** Al subir un RFP, se activa un endpoint de FastAPI que envía el contenido a Cohere API, la cual genera descripciones de roles y sus vectores de embeddings, que se almacenan en Supabase.

5. ***Cálculo de compatibilidad semántica:*** Una vez generados los roles, el sistema calcula la similaridad semántica entre el perfil del usuario y los roles activos, utilizando embeddings vectoriales y funciones RPC dentro de Supabase. Se presentan los porcentajes de compatibilidad visualmente.

6. ***Asignación de rol y visualización de reportes:*** El manager puede asignar manualmente un rol compatible a un usuario. El sistema actualiza el historial y genera reportes visibles en el dashboard, incluyendo KPIs individuales y grupales.

- **Diseño Lógico Clave:**

  El diseño lógico de la solución toma como base la arquitectura previamente descrita en su sección correspondiente, enfocándose en una composición modular y desacopla tanto en el cliente como en el backend.

  - ***Frontend (React + Vite):*** Cada vista del sistema está construida con un enfoque component-based, lo que permite una alta reusabilidad y mantenibilidad. Este diseño facilita una experiencia fluida, reactiva y escalable desde el lado del cliente. Se utilizan componentes reutilizables como:
    - Cards para mostrar proyectos, empleados o recomendaciones
    - Modales para inputs o confirmaciones
    - Tablas dinámicas para historial y datos relacionales
    - Gráficas para KPIs y compatibilidades
  - ***Backend desacoplado (FastAPI + Supabase):*** La lógica de negocio más compleja (como generación de contenido con IA o cálculo de compatibilidad) se delega a FastAPI, que actúa como coordinador entre el frontend, Supabase y Cohere. Supabase, como backend-as-a-service, centraliza persistencia, autenticación y almacenamiento sin necesidad de infraestructura personalizada adicional.
  - ***IA y eventos asincrónicos:*** La integración con Cohere API responde a eventos clave dentro de la plataforma (como la carga de un CV o un RFP). Estos eventos disparan flujos de procesamiento asincrónicos que devuelven resúmenes, roles o recomendaciones al usuario de manera transparente.

7. Vista de Procesos (RunTime)

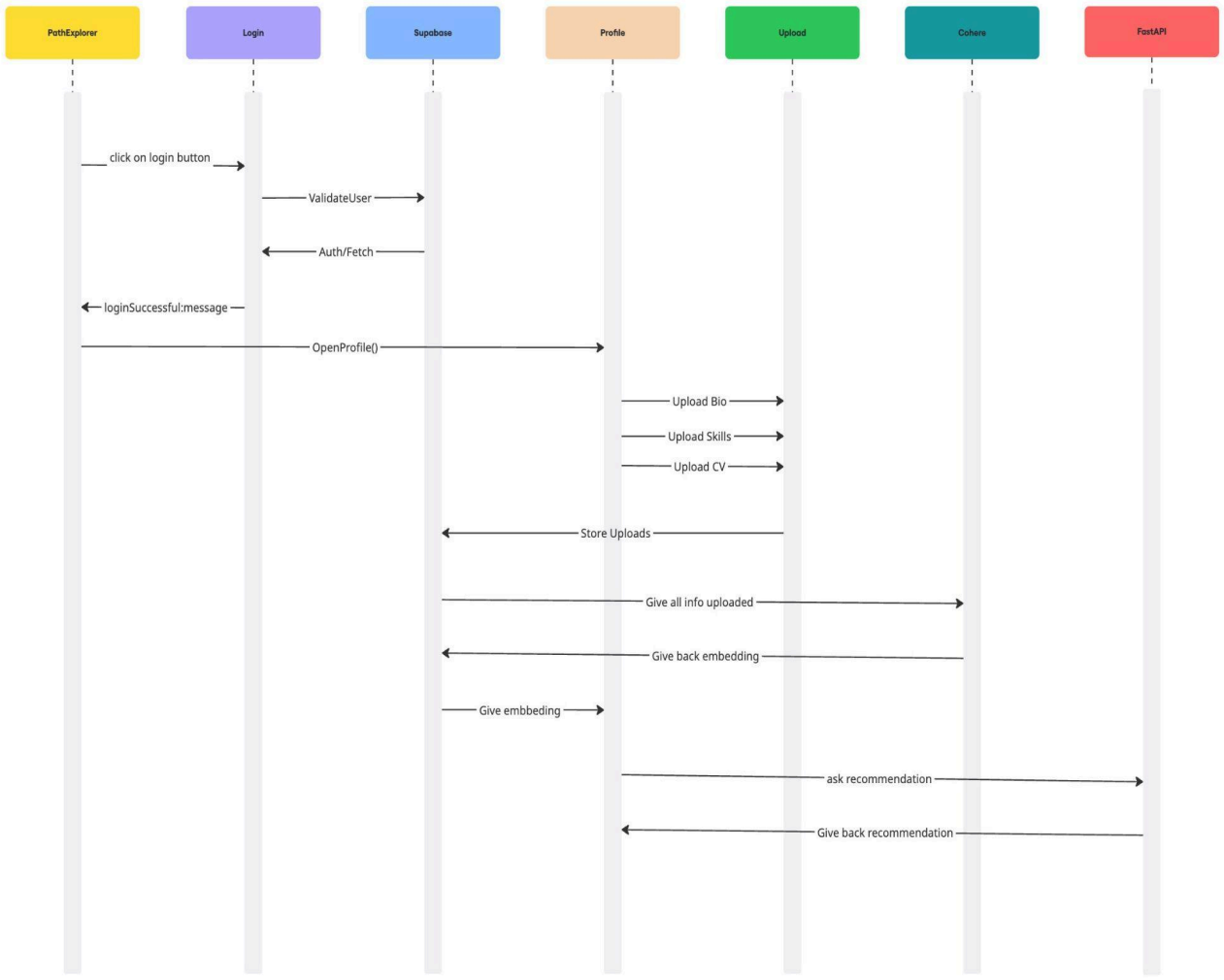
Escenario: Registro y Asignación

1. El usuario accede a la ruta pública / y posteriormente a /register para crear una cuenta.
2. Se realiza el alta del usuario mediante Supabase Auth. Una vez completado, se redirige al login (/login) y se autentica su sesión.
3. Tras autenticarse exitosamente, el usuario es redirigido a la vista /perfil, donde el sistema detecta información incompleta y solicita completar los datos.
4. El usuario completa su perfil subiendo:
   a. CV en .PDF → almacenado en Supabase Storage (cv/)
   b. Metas profesionales → almacenadas en la tabla Goal
   c. Habilidades → insertadas en la tabla relacional User_Skill
5. Un manager carga un documento RFP a través del frontend. El archivo es enviado a FastAPI, que lo procesa y lo reenvía a Cohere para la generación automática de roles (endpoint /generate-roles).

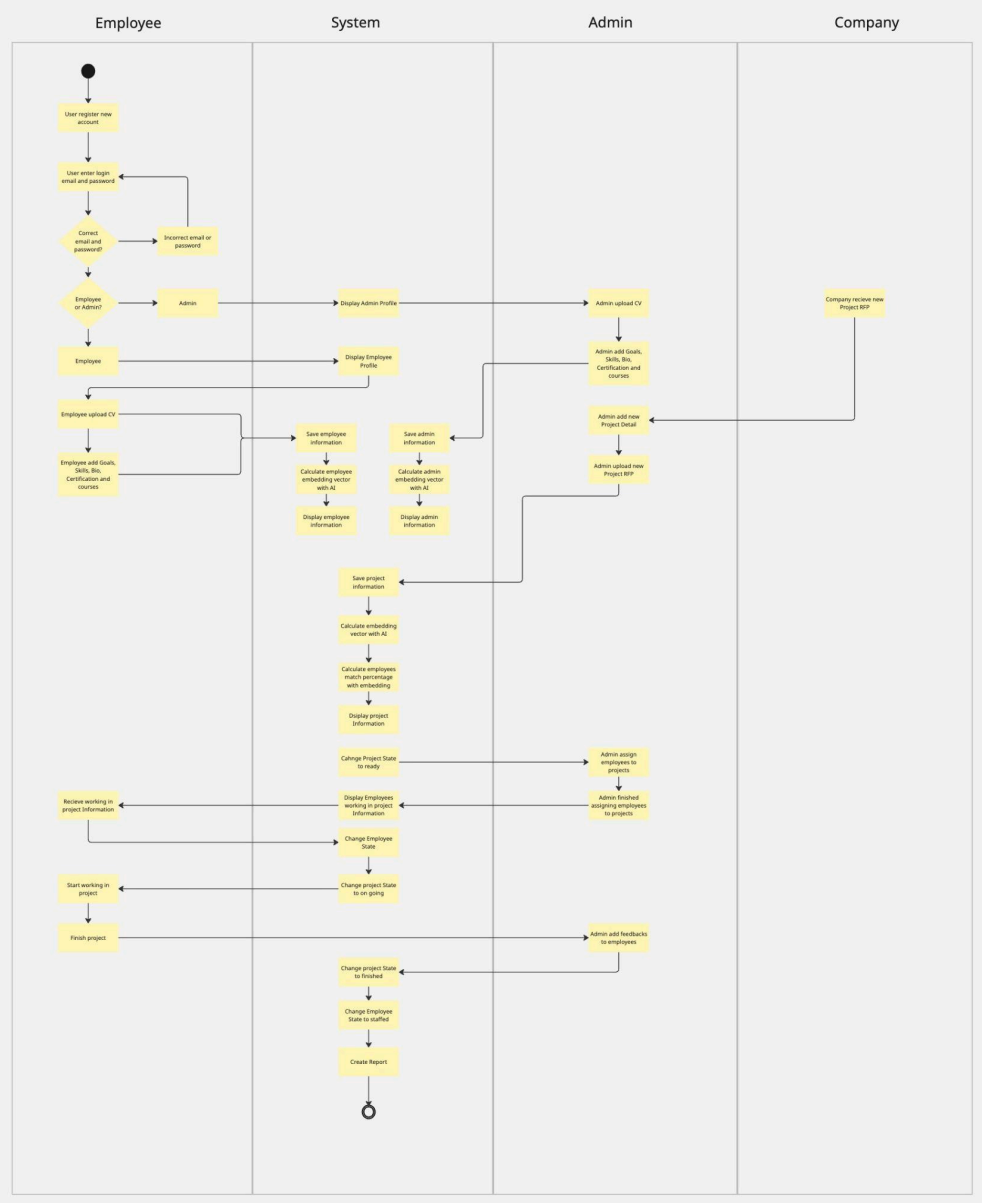
6. Una vez generados los roles, se calcula la similaridad entre los embeddings del perfil del usuario y los del rol. Este cálculo se realiza mediante un procedimiento remoto (RPC) en Supabase, y el porcentaje resultante se muestra en la interfaz.
7. El manager revisa los resultados y asigna manualmente un rol al usuario. Esta acción actualiza el estado en la tabla User_Role.
8. Se generan automáticamente reportes de asignación, métricas de compatibilidad y trayectoria profesional. Estos datos son visibles en el dashboard del usuario y del manager.

Diagramas:

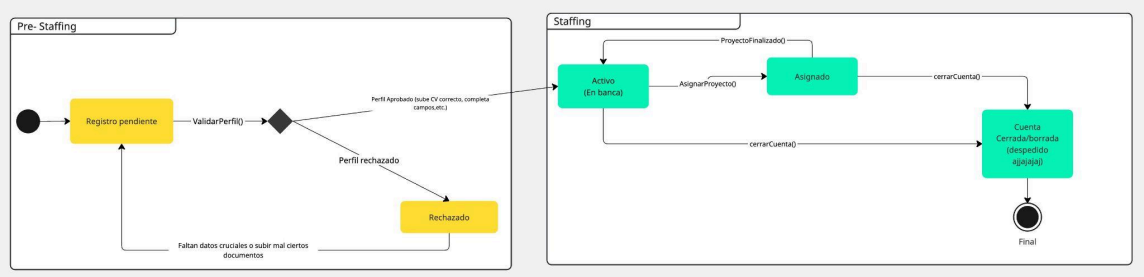
- Diagrama de secuencia (Login → Perfil → Upload → IA → Recomendación)



- Diagrama de actividad (flujo de uso de app)



- Diagrama de estados (Empleado: Inactivo → Activo → Asignado)



8. Rationale de la Arquitectura

La arquitectura seleccionada para PathExplorer responde a la necesidad de construir una solución escalable, modular y ágil, adecuada para entornos organizacionales dinámicos como el de Accenture. Las decisiones tomadas priorizan el desacoplamiento de responsabilidades, la facilidad de mantenimiento y la integración eficiente de capacidades inteligentes sin comprometer tiempos de entrega ni complejidad operativa.

Elecciones clave y su justificación:

- Se adoptó React por su capacidad para construir interfaces altamente interactivas mediante componentes reutilizables y su compatibilidad con un ecosistema de desarrollo ágil. Vite se eligió como bundler moderno por su velocidad en entornos de desarrollo y soporte nativo para módulos ES.
- La elección de Supabase permite acelerar el desarrollo al ofrecer funcionalidades de autenticación, persistencia estructurada (PostgreSQL) y almacenamiento de archivos sin necesidad de configurar servidores propios. Al tratarse de una solución gestionada, reduce la carga operativa del equipo y favorece un enfoque serverless para componentes no críticos.
- FastAPI cumple el rol de backend ligero enfocado en lógica de negocio especializada e integración con servicios externos. En particular, encapsula la comunicación con Cohere API, permitiendo incorporar capacidades de inteligencia artificial (generación de texto y embeddings semánticos) sin requerir modelos entrenados internamente, optimizando tiempo y recursos.
- El frontend se aloja en Vercel, lo que permite despliegues continuos automáticos con control de versiones. FastAPI se ejecuta en una máquina virtual Linux, y todos los archivos generados o subidos por el usuario se almacenan en Supabase Storage, consolidando así un stack completamente gestionado y escalable con bajo costo operativo.

Alternativas descartadas:

- Se descartó una arquitectura monolítica tradicional debido a sus limitaciones en escalabilidad, complejidad en el mantenimiento y tiempos prolongados de implementación.
- Se evitó configurar un backend autogestionado completo para autenticación, base de datos o almacenamiento, ya que hubiese requerido recursos adicionales no proporcionales al alcance del proyecto.