Definición del problema:

Las áreas de recursos humanos enfrentan problemas al momento de atraer nuevo talento. Los procesos son lentos, costosos, ineficientes y muchas tareas se realizan de forma manual. Desde las fases iniciales de contratación suele haber muchos procesos repetitivos e ineficientes que hacen que el proceso de contratación se ralentice y en cuantiosas ocasiones que estos procesos se caigan. Este problema afecta a los interesados puesto que tanto empresa como aspirantes sufren de estas ineficiencias.

Solución:

Una aplicación web que centraliza y automatiza las etapas básicas del proceso de contratación. La web permitirá recibir aplicaciones en un solo lugar, evaluar candidatos rápidamente, dar seguimiento dentro de la misma web al empleado. Usar IA para analizar los CVS y ver cual se ajusta mas cierta plaza y que el sistema los ordene.

Funciones específicas:

- Publicación de vacantes: RRHH puede crear vacantes con titulo, descripción, requisitos, salario estimado.
- Empleado puede llenar formulario web donde puede poner su nombre, correo, cargar CV desde la pc. Puesto al que aplica, pretensión salarial, experiencia.
- Un dashboard donde RRHH puede ver la lista de empleados por plaza, y un sistema de IA, pondrá en las primeras posiciones a las plazas que mas se ajusten a dicho perfil.
- RRHH puede cargar pruebas al sistema, y estas le parecerán al usuario.
- Chat directo de la persona de RRHH

Justificación: La digitalización de procesos de contratación, es una necesidad en empresas de todos los tamaños. Los métodos tradicionales basados en hojas de calculo, correos electrónicos, hojas de cálculo y revisión manual generan una gran carga operativa a los equipos de recursos humanos. La solución presentada aporta valor al centralizar los elementos del proceso de contratación, mejora drásticamente los tiempos de contratación y la experiencia del candidato/reclutador.

FrontEnd

Comparación de tecnologías:

React: Biblioteca flexible y altamente adoptada. Cuenta con una gran comunidad, por lo que hallar documentación es relativamente fácil. Cuenta con un amplio ecosistema de herramientas y componentes que ayudan a desplegar interfaces.

Vue.js: Tiene una sintexis simple, y una curva de aprendizaje baja. Es una comunidad mas pequeña respecto a react. Es ideal para proyectoes pequeños y que requieran un despliegue rápido.

Angular: De los 3 es probablemente el framework mas robusto. Esto hace que su curva de aprendizaje también sea mas alta que los otros. Es ideal para proyectos bastante grandes, que requieran un gran despliegue.

Eleccion final:

La elección final será React. Por su curva de aprendizaje moderada, su capacidad de desplegar interfaces completas.

Backend:

Java+Spring = Este es un stack bastante robusto para aplicaciones empresariales grandes y complejas. Puede ser bastante pesado para un prototipo simple. Tiene múltiples funcionalidades, respaldado por mucha documentación.

Node.js + Express = Es ligero rápido, y orientado al uso de Apis Rest. Cuenta con un ecosistema amplio y fácil de desplegar. El uso de json nativo hace que la comunicación con el frontend sea relativamente fácil de implementar

Eleccion final: Node.js + Express.

El combo react con Node.js + Express permite desplegar un desarrollo web completo en poco tiempo gracias a a su flexibilidad, integración con servicios Rest además de compatibilidad con despliegues económicos o gratuitos.

Para la publicación del servicio, se utilizará render, ya que ofrece un plan gratuito. Y su configuración inicial es sencilla.

Persistencia de datos

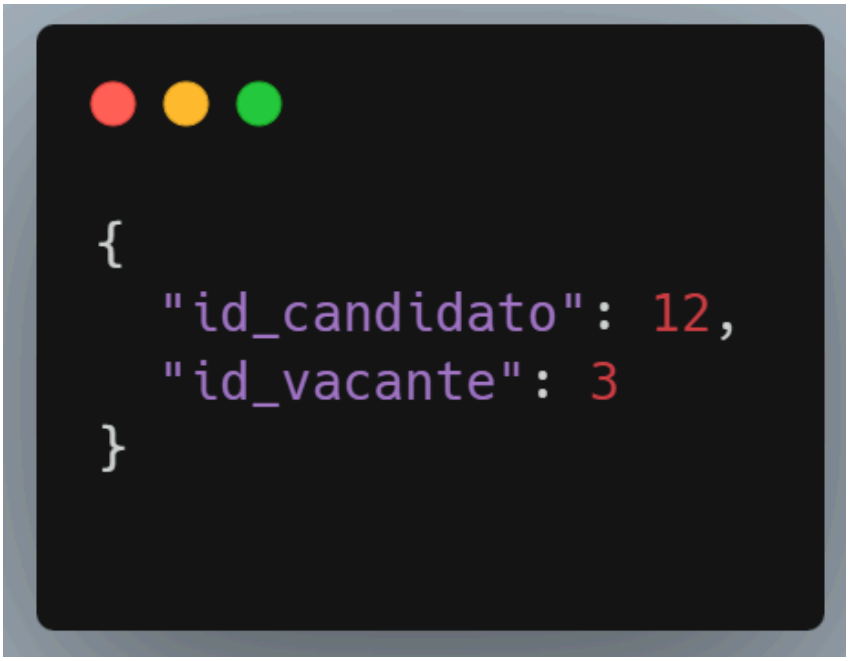
SQL vs Nosql

El sistema guardara información estructurada y relacionada entre si por lo que es necesario usar bases de datos SQl. PostgreSQL se elige por ser robusto, gratuito, soportar claves foráneas, joins y escalabilidad.

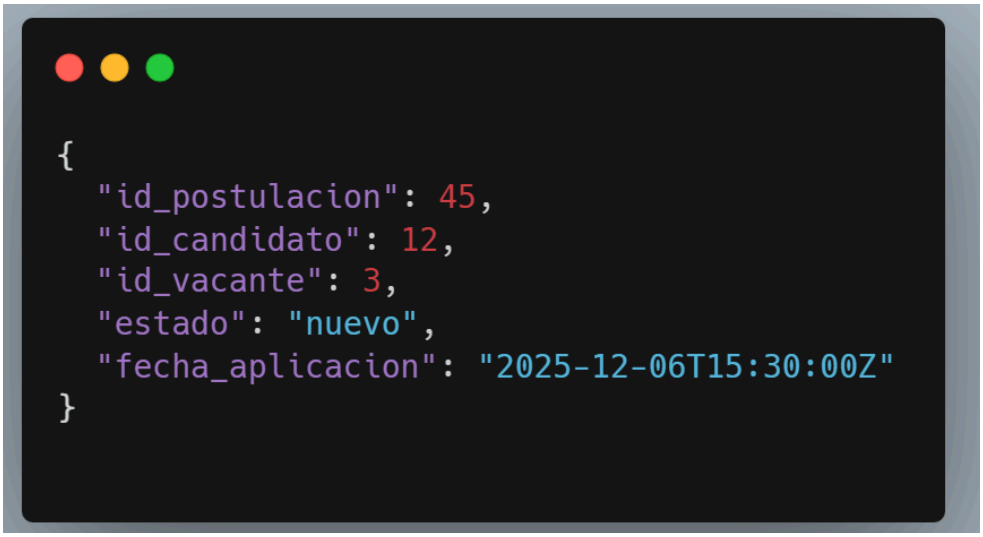
Generación de endpoints

ENDPOINT 1 — Crear Postulación (El candidato aplica a una vacante)

- **Método:** POST
- **Ruta:** /api/v1/crearPostulacion
- **Descripción:** El frontend envía los datos del candidato y la vacante seleccionada. El backend registra la postulación con estado inicial *"nuevo"*.
- **Body:**



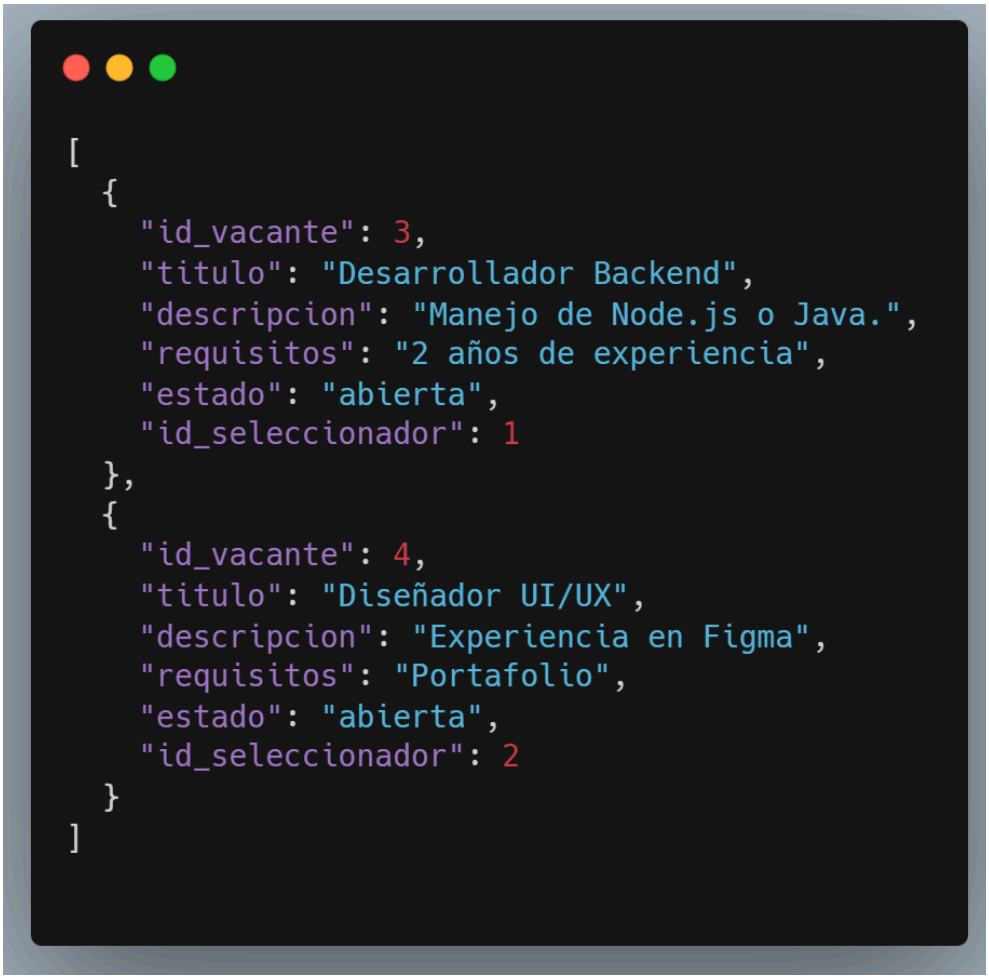
Respuesta Exitosa (201 Created):



```
{
  "id_postulacion": 45,
  "id_candidato": 12,
  "id_vacante": 3,
  "estado": "nuevo",
  "fecha_aplicacion": "2025-12-06T15:30:00Z"
}
```

ENDPOINT 2 — Obtener Vacantes Disponibles

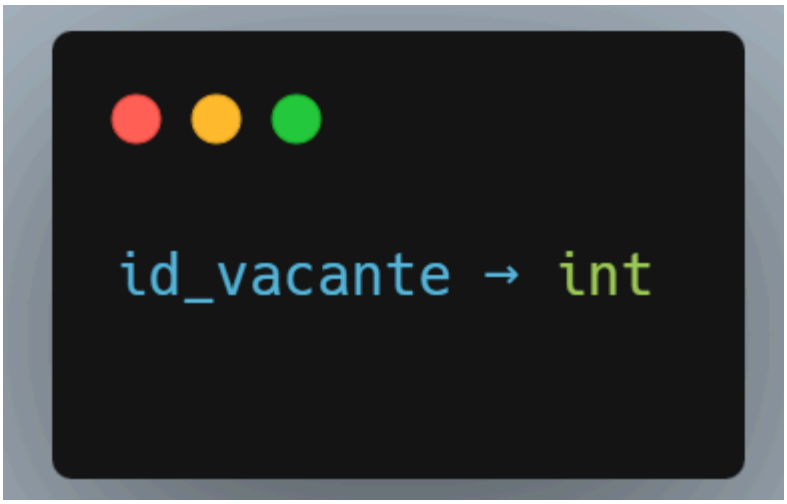
- **Método:** GET
- **Ruta:** /api/v1/verPostulaciones
- **Descripción:**El frontend utiliza este endpoint para mostrar al candidato o al seleccionador todas las vacantes disponibles en el sistema.
- **Respuesta (200 OK):**




```
[
  {
    "id_vacante": 3,
    "titulo": "Desarrollador Backend",
    "descripcion": "Manejo de Node.js o Java.",
    "requisitos": "2 años de experiencia",
    "estado": "abierta",
    "id_seleccionador": 1
  },
  {
    "id_vacante": 4,
    "titulo": "Diseñador UI/UX",
    "descripcion": "Experiencia en Figma",
    "requisitos": "Portafolio",
    "estado": "abierta",
    "id_seleccionador": 2
  }
]
```


ENDPOINT 3 — Obtener Pruebas por Vacante

- **Método:** GET
- **Ruta:** /api/vacantes/{id_vacante}/pruebas
- **Descripción:**Cuando el candidato abre una vacante en el frontend, este endpoint obtiene la lista de pruebas asociadas.
- **Parámetro de ruta:**

- 

- **Respuesta (200 OK):**