INSTITUTO PROFESIONAL SAN SEBASTIAN

DESARROLLO BACKEND

EVALUACION n°2

PROFESOR: VICENTE ZAPATA

ALUMNO: CESAR RUBILAR

Documentación de la API – Evaluación Unidad 2

Este documento detalla los endpoints implementados para el perfil de Reclutador, enfocados en la gestión de la entidad **oferta_laboral**. Se cumplen los requisitos mínimos establecidos, incluyendo los métodos HTTP:

POST, **GET**, **PUT**, **PATCH** (para actualización de estado)

PATCH (para simulación de eliminación).

Crear oferta laboral

- **Método:** POST
- **URL:** /api/ofertas
- **Descripción:** Permite crear una nueva oferta laboral con estado inicial **"activa"**.

Ejemplo de Request Body:

json

```
{

  "titulo": "Desarrollador Backend",
  "descripcion": "Desarrollo con Node.js",
  "requisitos": "3 años de experiencia",
  "salario": 1000000,
  "fecha_publicacion": "2025-05-18"
}
```

Ejemplo de Respuesta:

json

```
{

  "mensaje": "Oferta laboral creada exitosamente",
  "id": 1
}
```

Listar todas las ofertas

- **Método:** GET
- **URL:** /api/ofertas
- **Descripción:** Recupera el listado completo de ofertas laborales registradas.
- **Nota:** No requiere enviar body en la petición.

Ejemplo de Respuesta:

json

```
[
  {
    "id": 1,
    "titulo": "Desarrollador Backend",
    "estado": "activa"
  }
]
```

Editar una oferta

- **Método:** PUT
- **URL:** /api/ofertas/:id
- **Descripción:** Actualiza la información de una oferta laboral existente identificada por su ID.

Ejemplo de Request Body:

json

```
{
  "titulo": "Desarrollador Fullstack",
  "descripcion": "Backend y frontend",
  "requisitos": "Experiencia con Node.js y Vue",
  "salario": 1200000,
  "fecha_publicacion": "2025-05-20"
```

}

Ejemplo de Respuesta:

json

{

"mensaje": "Oferta laboral actualizada correctamente"

}

Cambiar estado de la oferta

- **Método:** PATCH
- **URL:** /api/ofertas/estado/:id
- **Descripción:** Modifica el estado de una oferta laboral (por ejemplo, de **activa** a **inactiva**).

Ejemplo de Request Body:

json

{

"estado": "inactiva"

}

Ejemplo de Respuesta:

json

{

"mensaje": "Estado actualizado a inactiva"

}

Simular eliminación de oferta

- **Método:** PATCH
- **URL:** /api/ofertas/remove/:id
- **Descripción:** Realiza una desactivación lógica de la oferta, simulando su eliminación.
- **Nota:** No requiere body en la petición.

Ejemplo de Respuesta:

json

```
{
  "mensaje": "Oferta laboral desactivada (simulación de eliminación)"
}
```

Informe Evaluación Backend – Unidad 2

CE1: Diseño de la API

Se desarrolló un conjunto de endpoints RESTful centrados en la entidad **oferta_laboral**, orientados al perfil de Reclutador. La API incluye los métodos HTTP **POST**, **GET**, **PUT** y **PATCH**. Además, se entregó documentación técnica detallada en un documento aparte, con ejemplos claros de requests y responses.

CE2: Integración de la API

Se configuró el archivo app.js para integrar los endpoints bajo el prefijo /api/ofertas, utilizando Express Router. Todos los endpoints están correctamente vinculados a sus respectivos controladores, garantizando una estructura modular y mantenible.

CE3: Mantenimiento de la API

Los controladores fueron implementados siguiendo buenas prácticas de desarrollo. Se incorporó validación básica de los datos requeridos para la creación y edición de ofertas laborales. Asimismo, se incluyó un manejo adecuado de errores, con respuestas estructuradas en formato JSON.

CE4: Actualización de la API

Se llevaron a cabo pruebas de actualización mediante el método **PUT** para modificar registros completos y **PATCH** para cambios puntuales, como la actualización del estado de una oferta. Se aseguró la compatibilidad con versiones anteriores mediante convenciones claras en las rutas.

CE5: Desarrollo del servicio web RESTful

La API fue desarrollada con **Node.js** y **Express**. Las pruebas de funcionamiento se realizaron utilizando **Thunder Client**, simulando el consumo de cada endpoint definido. Se verificaron respuestas esperadas con códigos de estado HTTP adecuados.

CE6: Colección de Thunder Client

Se creó una colección de pruebas en Thunder Client que incluye las cinco rutas implementadas. Esta colección fue exportada en formato .json y entregada junto al proyecto para facilitar su revisión y replicación de pruebas.

CE7: Operaciones CRUD

Se implementó el ciclo completo de operaciones CRUD sobre la tabla **oferta_laboral**. Estas incluyen creación, consulta, edición, cambio de estado y desactivación lógica de registros. Se garantizó la integridad de los datos mediante consultas preparadas y manejo de errores.

CE8: Pruebas y validación

En línea con el alcance de esta evaluación, se realizaron pruebas manuales con Thunder Client, simulando diversos escenarios. Esto incluyó pruebas de creación con campos faltantes para validar la robustez de las validaciones, así como respuestas controladas ante errores del servidor.