**Reto Técnico para el Perfil de Backend**

Postulante: Danery Misshel Santos Alcarraz

Fecha: 22/09/2024

**ÍNDICE**

[1. Código fuente versionado 3](#_Toc177870889)

[2. Documentación de los trabajos realizados 3](#_Toc177870890)

[A. Modelo Entidad – Relación 3](#_Toc177870891)

[B. Modelo Lógico 3](#_Toc177870892)

[C. Diccionario de Datos 4](#_Toc177870893)

[3. Inventario de librerías y componentes y detalles sobre la base de datos 5](#_Toc177870894)

[A. Librerías (definidas en el archivo pom.xml) usadas para la construcción de los servicios 6](#_Toc177870895)

[B. Componentes usados para la construcción de los servicios 7](#_Toc177870896)

[C. Detalles de la base de datos seleccionada para almacenar la información 9](#_Toc177870897)

[4. Catálogo de los servicios web implementados, incluyendo el detalle de los parámetros usados para la invocación 10](#_Toc177870898)

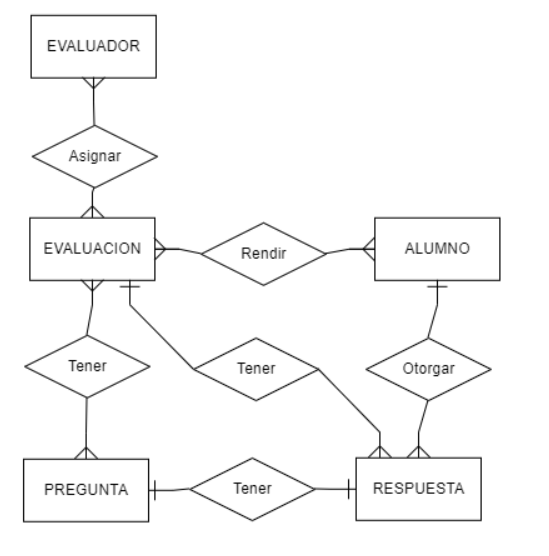
[5. Recomendaciones 16](#_Toc177870899)

# **Código fuente versionado**

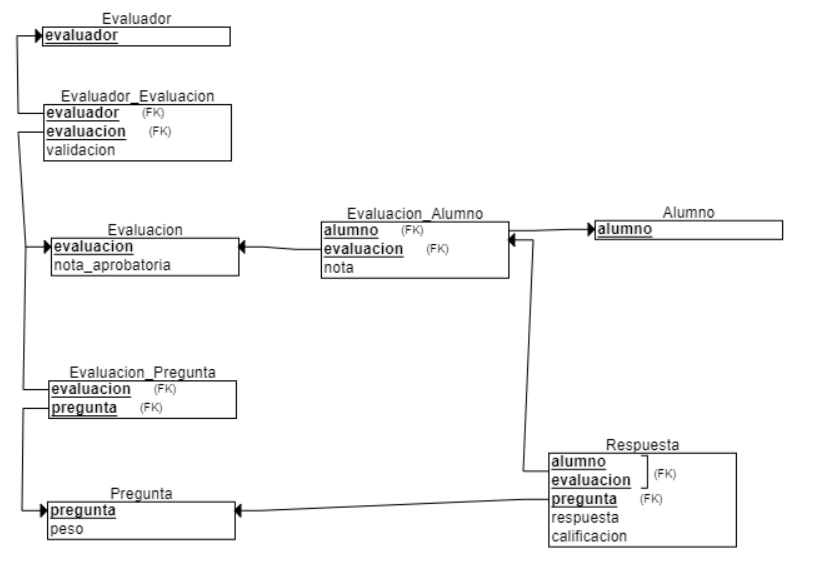
# **Documentación de los trabajos realizados**

Software usado: ERDPlus

## **Modelo Entidad – Relación**

****

## **Modelo Lógico**

****

## **Diccionario de Datos**

**Tabla: Alumno**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| alumno | INT | Identificador único para cada alumno (PK). |

**Tabla: Pregunta**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| pregunta | VARCHAR(255) | Texto descriptivo de la pregunta (PK). |
| peso | INT | Peso asignado a la pregunta en una evaluación. |

**Tabla: Evaluador**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| evaluador | INT | Identificador único para cada evaluador (PK). |

**Tabla: Evaluacion**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| evaluacion | INT | Identificador único para cada evaluación (PK). |
| nota\_aprobatoria | INT | Nota mínima requerida para aprobar la evaluación. |

**Tabla: Evaluacion\_Pregunta**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| evaluacion | INT | Identificador único para cada evaluación (PK, FK). |
| pregunta | VARCHAR(255) | Texto descriptivo de la pregunta (PK, FK). |

**Tabla: Evaluacion\_Alumno**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| nota | INT | Nota final obtenida por el alumno en la evaluación. |
| alumno | INT | Identificador del alumno (PK, FK). |
| evaluacion | INT | Identificador de la evaluación (PK, FK). |

**Tabla: Respuesta**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| alumno | INT | Identificador único para cada alumno (PK, FK). |
| evaluacion | INT | Identificador único para cada evaluación (PK, FK). |
| pregunta | VARCHAR(255) | Texto descriptivo de la pregunta (PK, FK). |
| respuesta | VARCHAR(255) | Respuesta dada por el alumno. |
| calificacion | INT | Calificación asignada por el evaluador. |

**Tabla: Evaluador\_Evaluacion**

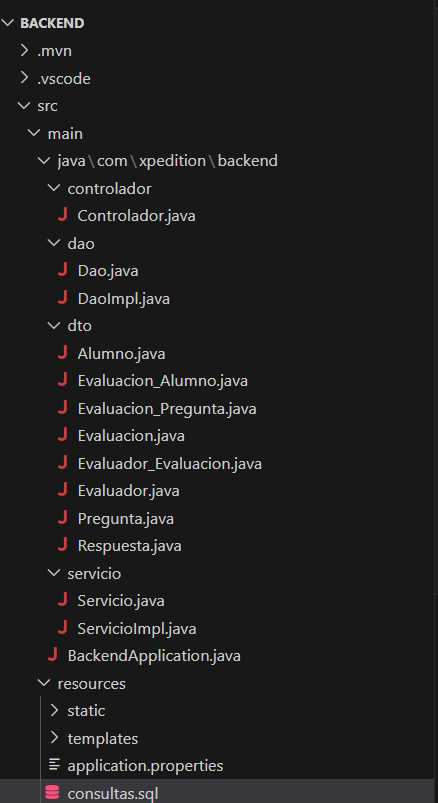
|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| validacion | INT | Valor que indica si el evaluador está autorizado para calificar la evaluación. Si es 1, está autorizado, si es 0, no está autorizado. |
| evaluador | INT | Identificador único para cada evaluador (PK, FK). |
| evaluacion | INT | Identificador único para cada evaluación (PK, FK). |

# **Inventario de librerías y componentes y detalles sobre la base de datos**

## **Librerías (definidas en el archivo pom.xml) usadas para la construcción de los servicios**

1. Spring Boot Starter Data JPA:
   * Grupo: org.springframework.boot
   * Artefacto: spring-boot-starter-data-jpa
   * Descripción: Proporciona soporte para JPA (Java Persistence API) que permite interactuar con la base de datos de manera sencilla y eficiente.
2. Spring Boot Starter Web:
   * Grupo: org.springframework.boot
   * Artefacto: spring-boot-starter-web
   * Descripción: Permite la creación de aplicaciones web, proporcionando un servidor embebido (Tomcat por defecto) y soporte para controladores REST.
3. PostgreSQL:
   * Grupo: org.postgresql
   * Artefacto: postgresql
   * Versión: 42.7.4
   * Descripción: Driver JDBC para interactuar con bases de datos PostgreSQL.
4. Lombok:
   * Grupo: org.projectlombok
   * Artefacto: lombok
   * Versión: 1.18.32
   * Descripción: Herramienta para eliminar código boilerplate en las clases Java (como getters, setters, constructores, etc.).
5. Jackson Databind:
   * Grupo: com.fasterxml.jackson.core
   * Artefacto: jackson-databind
   * Descripción: Librería que permite la serialización y deserialización de objetos Java a JSON y viceversa.

## **Componentes usados para la construcción de los servicios**



**Controlador (Controlador.java)**

* Encargado de manejar las peticiones HTTP y redirigirlas a los métodos correspondientes del servicio.
* Endpoints implementados:
  + /insertar-respuestas: Inserta las respuestas de los alumnos.
  + /obtener-respuesta: Obtiene una respuesta específica.
  + /calificar-respuesta: Califica una respuesta de un alumno.
  + /configurar-peso-pregunta: Configura el peso de una pregunta en una evaluación.
  + /configurar-nota-aprobatoria: Configura la nota aprobatoria de una evaluación.
  + /obtener-respuestas-alumno: Obtiene las respuestas de un alumno, su nota y si ha aprobado.

**DAO (Dao.java)**

* Interfaz que define las operaciones de acceso a datos.

**DAO Implementación (DaoImpl.java)**

* Implementa la interfaz Dao y realiza las operaciones en la base de datos utilizando JdbcTemplate.
* Métodos implementados:
  + insertarRespuestas: Inserta las respuestas de un alumno en una evaluación.
  + obtenerRespuesta: Obtiene una respuesta específica según una alumno haya respondido a una pregunta a una evaluación.
  + esEvaluadorValidado: Valida si un evaluador está autorizado.
  + calificarRespuesta: Califica una respuesta por un evaluador autorizado.
  + actualizarPesoPregunta: Actualiza el peso de una pregunta en una evaluación.
  + actualizarNotaAprobatoria: Actualiza la nota aprobatoria de una evaluación.
  + obtenerRespuestasAlumno: Obtiene las respuestas de un alumno y si aprobó la evaluación que rindió.
  + obtenerNotaAlumno: Obtiene la nota de un alumno.
  + obtenerNotaAprobatoria: Obtiene la nota aprobatoria de una evaluación.

**Servicio (Servicio.java)**

* Interfaz que define la lógica de negocio.

**Servicio Implementación (ServicioImpl.java)**

* Implementa la interfaz Servicio y delega las operaciones al DAO.
* Métodos implementados:
  + insertarRespuestas: Llama al DAO para insertar las respuestas.
  + obtenerRespuesta: Llama al DAO para obtener una respuesta.
  + validarEvaluador: Valida si un evaluador está autorizado.
  + calificarRespuesta: Llama al DAO para calificar una respuesta.
  + actualizarPesoPregunta: Llama al DAO para actualizar el peso de una pregunta.
  + actualizarNotaAprobatoria: Llama al DAO para actualizar la nota aprobatoria.
  + obtenerRespuestasAlumno: Llama al DAO para obtener las respuestas de un alumno.
  + obtenerNotaAlumno: Llama al DAO para obtener la nota de un alumno.
  + obtenerNotaAprobatoria: Llama al DAO para obtener la nota aprobatoria.
  + esAprobado: Calcula si un alumno ha aprobado o no según su nota.

**DTOs (Folder dto)**

* Entidades mapeadas a las tablas de la base de datos, usadas para transportar la información entre capas.
* **Tablas en la base de datos:**
  + Alumno: Contiene los datos de los alumnos.
  + Pregunta: Define las preguntas y su peso.
  + Evaluador: Contiene los evaluadores.
  + Evaluacion: Define las evaluaciones y la nota aprobatoria.
  + Evaluacion\_Pregunta: Relaciona evaluaciones con preguntas.
  + Evaluacion\_Alumno: Relaciona alumnos con evaluaciones y contiene la nota de los alumnos.
  + Respuesta: Almacena las respuestas de los alumnos.
  + Evaluador\_Evaluacion: Relaciona los evaluadores con las evaluaciones y contiene la validación.

## **Detalles de la base de datos seleccionada para almacenar la información**

* Motor de base de datos: PostgreSQL
* Versión del driver: 42.7.4
* Configuración de conexión (definida en application.properties):
  + URL: jdbc:postgresql://localhost:5432/xpedition
  + Usuario: postgres
  + Contraseña: postgres
  + Configuración adicional:
    - spring.jpa.hibernate.ddl-auto=update: Hibernate gestionará el esquema de la base de datos.
    - spring.jpa.show-sql=true: Muestra las consultas SQL ejecutadas en la consola.

Cabe resaltar que en el Folder src/main/resources puede encontrar un archivo consultas.sql para las inserciones necesarias para ejecutar el código.

# **Catálogo de los servicios web implementados, incluyendo el detalle de los parámetros usados para la invocación**

1. **Insertar respuestas:** Inserta un conjunto de respuestas de un alumno que rindió una evaluación en la base de datos. Método HTTP POST.

**Endpoint**: /insertar-respuestas

**Parámetros**:

* **Body (JSON)**: Lista de mapas con respuestas en formato:

[

{

"alumno": 101020,

"evaluación": 20200033,

"respuestas": [

{

"pregunta\_1": "opcion a",

"pregunta\_2": "<p> Fames rutrum ultricies nunc donec <strong> vivamus euismod volutpat interdum senectus </strong>. </p>",

"pregunta\_3": "1; 3; 5",

"pregunta\_4": "Arcu phasellus bibendum eget cum, nascetur dignissim venenatis class",

"pregunta\_5": "def sort\_colors\_by\_hex(colors): return sorted(colors, key=lambda color: int(color.lstrip('#'), 16))"

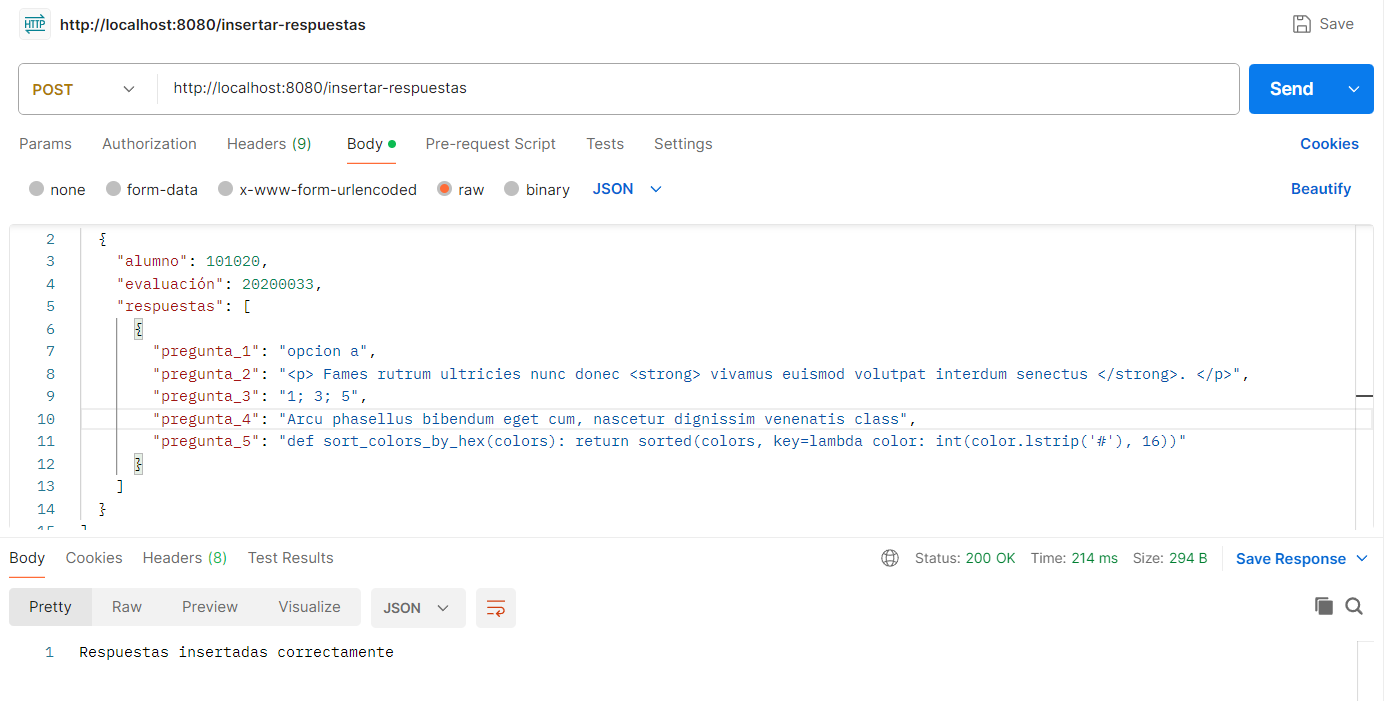
}

]

}

]

**Resultado:**

****

1. **Obtener respuesta:** Obtiene la respuesta específica de un alumno para una evaluación y pregunta. Método HTTP POST.

**Endpoint**: /obtener-respuesta

**Parámetros**:

* **Body (JSON)**:

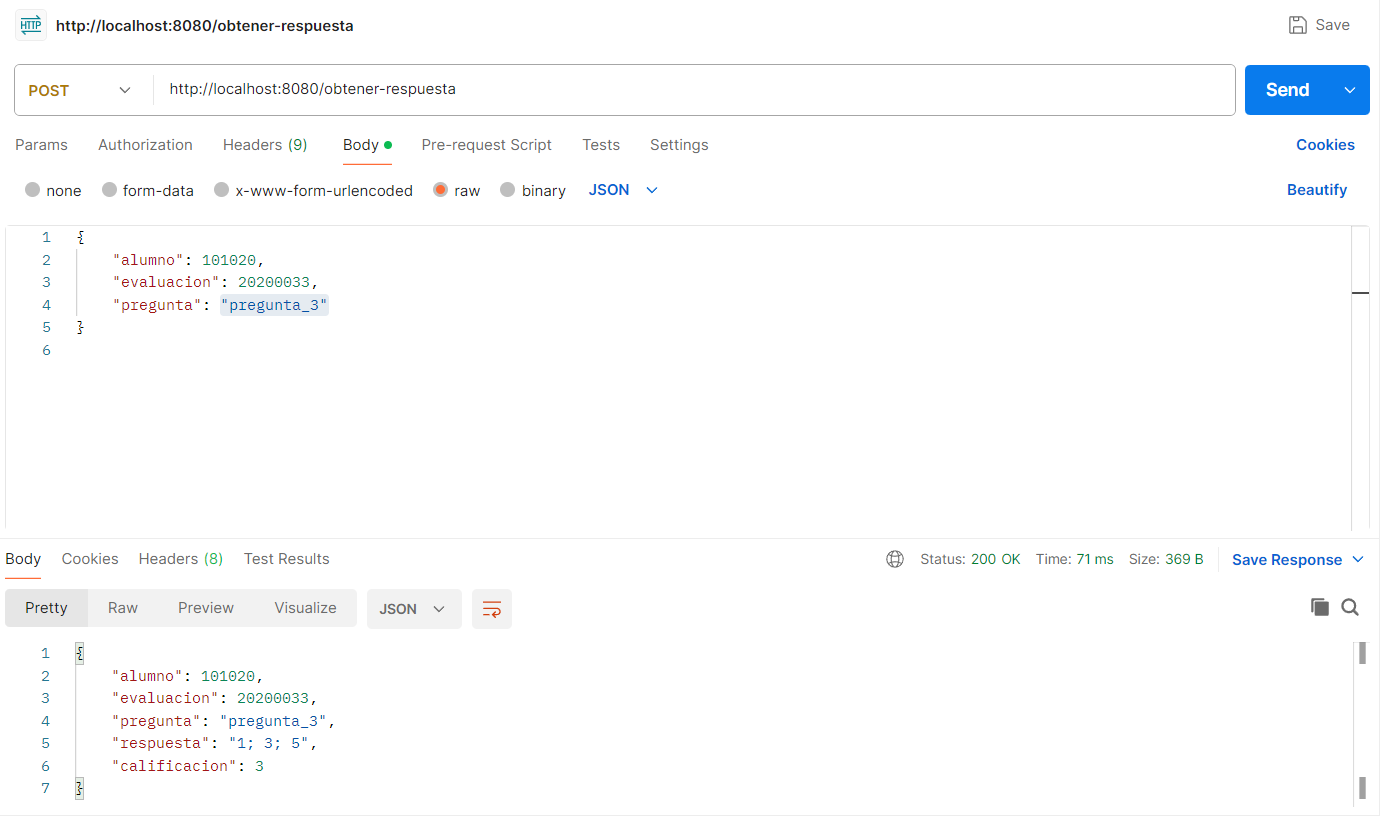
{

"alumno": 101020,

"evaluacion": 20200033,

"pregunta": "pregunta\_3"

}

**Resultado:  
**

1. **Calificar respuesta:** Permite al evaluador calificar una respuesta de un alumno, validando si el evaluador está autorizado. Método HTTP POST.

**Endpoint**: /calificar-respuesta

**Parámetros**:

* **Body (JSON)**:

{

"evaluador": 1,

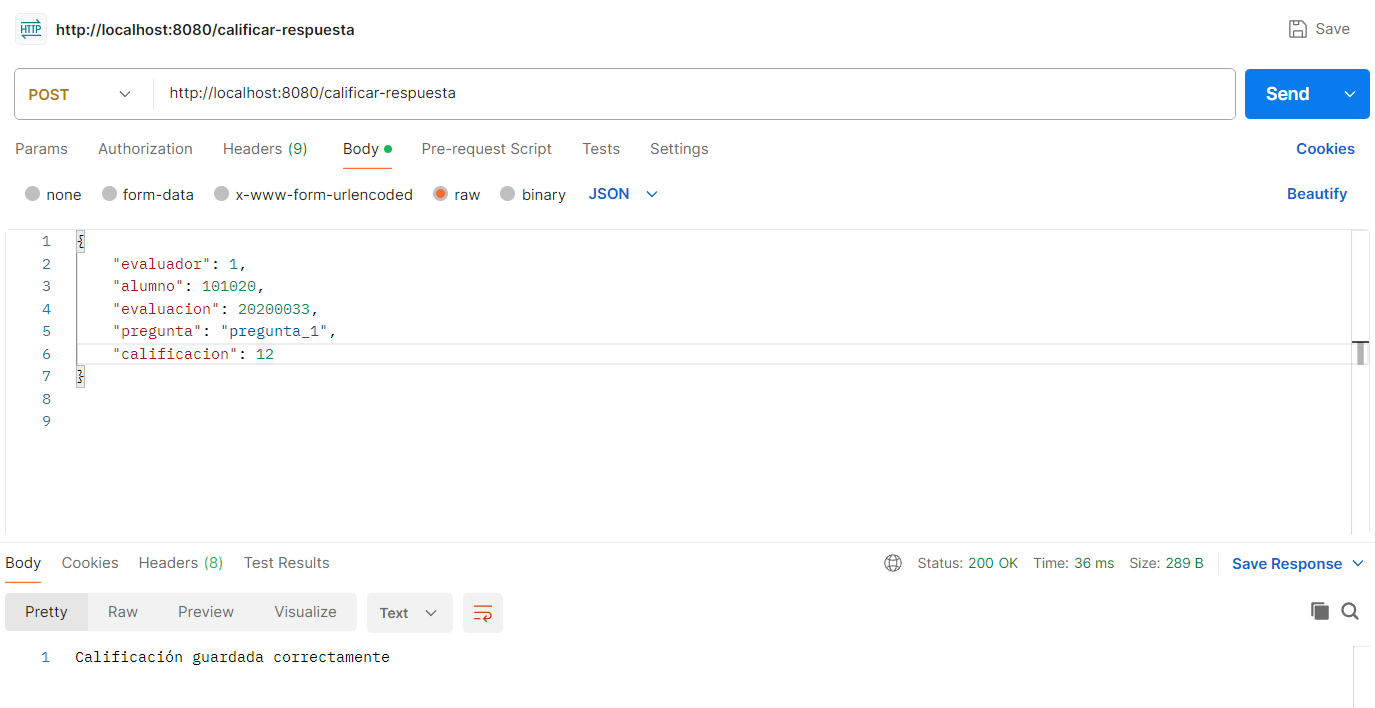
"alumno": 101020,

"evaluacion": 20200033,

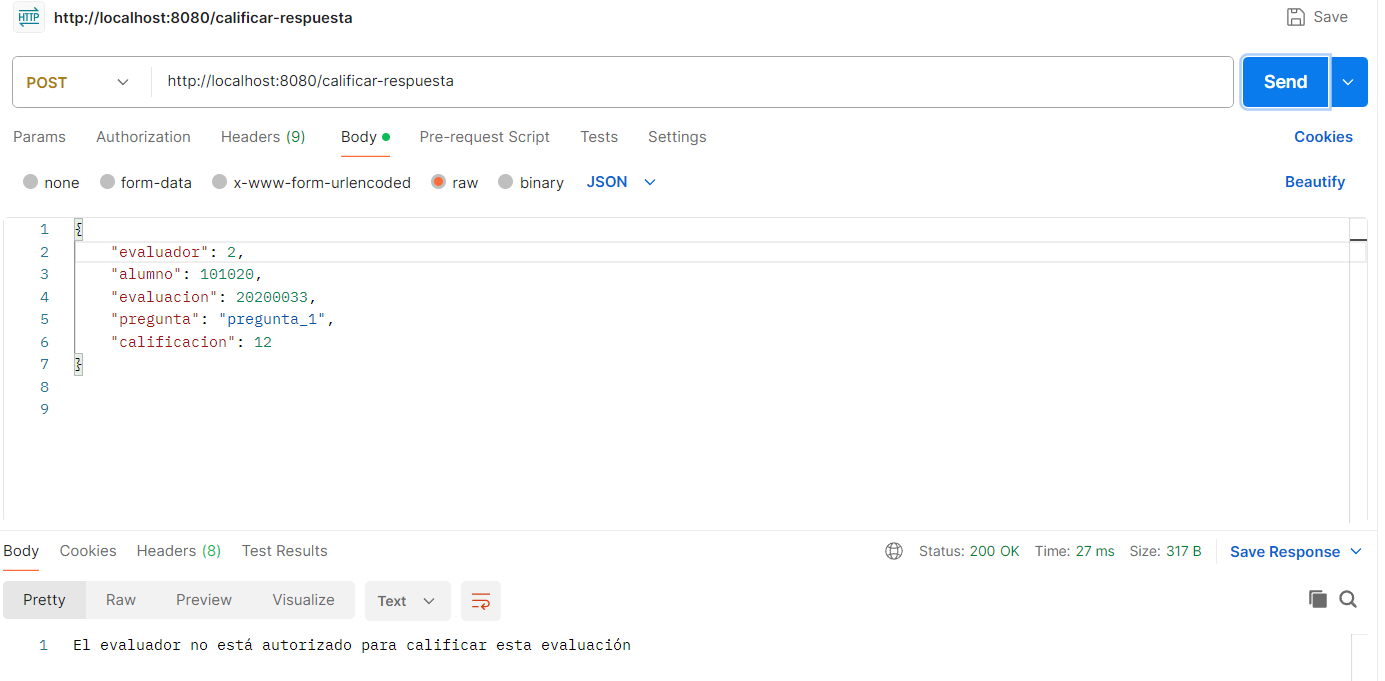
"pregunta": "pregunta\_1",

"calificacion": 12

}

**Respuesta:**- Cuando es un evaluador autorizado (atributo validación = 1 en la tabla Evaluador\_Evaluacion): ****

- Cuando es un evaluador no autorizado (atributo validación = 0 en la tabla Evaluador\_Evaluacion):

****

1. **Configurar peso de pregunta:** Configura el peso de una pregunta en una evaluación. Método HTTP POST.

**Endpoint**: /configurar-peso-pregunta

**Parámetros**:

* **Body (JSON)**:

{

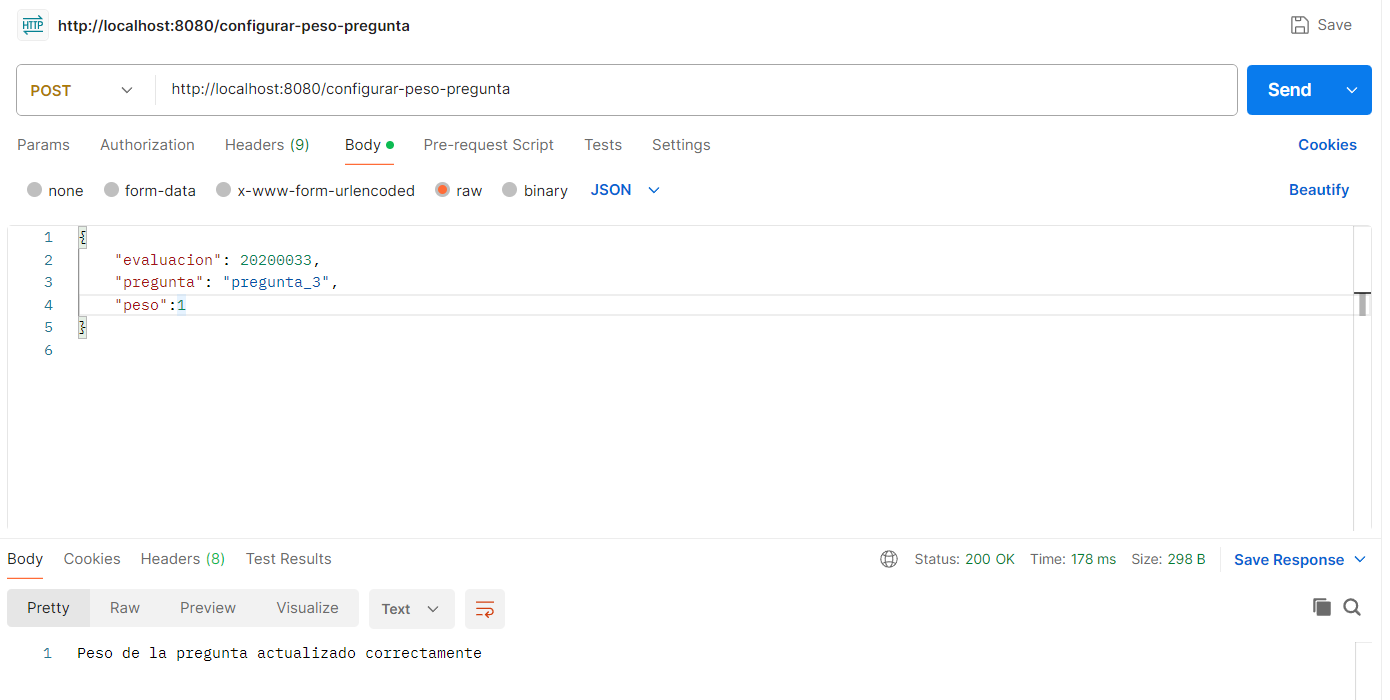
"evaluacion": 20200033,

"pregunta": "pregunta\_3",

"peso":1

}

**Respuesta:**

****

1. **Configurar nota aprobatoria:** Configura la nota mínima aprobatoria para una evaluación. Método HTTP POST.

**Endpoint**: /configurar-nota-aprobatoria

**Parámetros**:

* **Body (JSON)**:

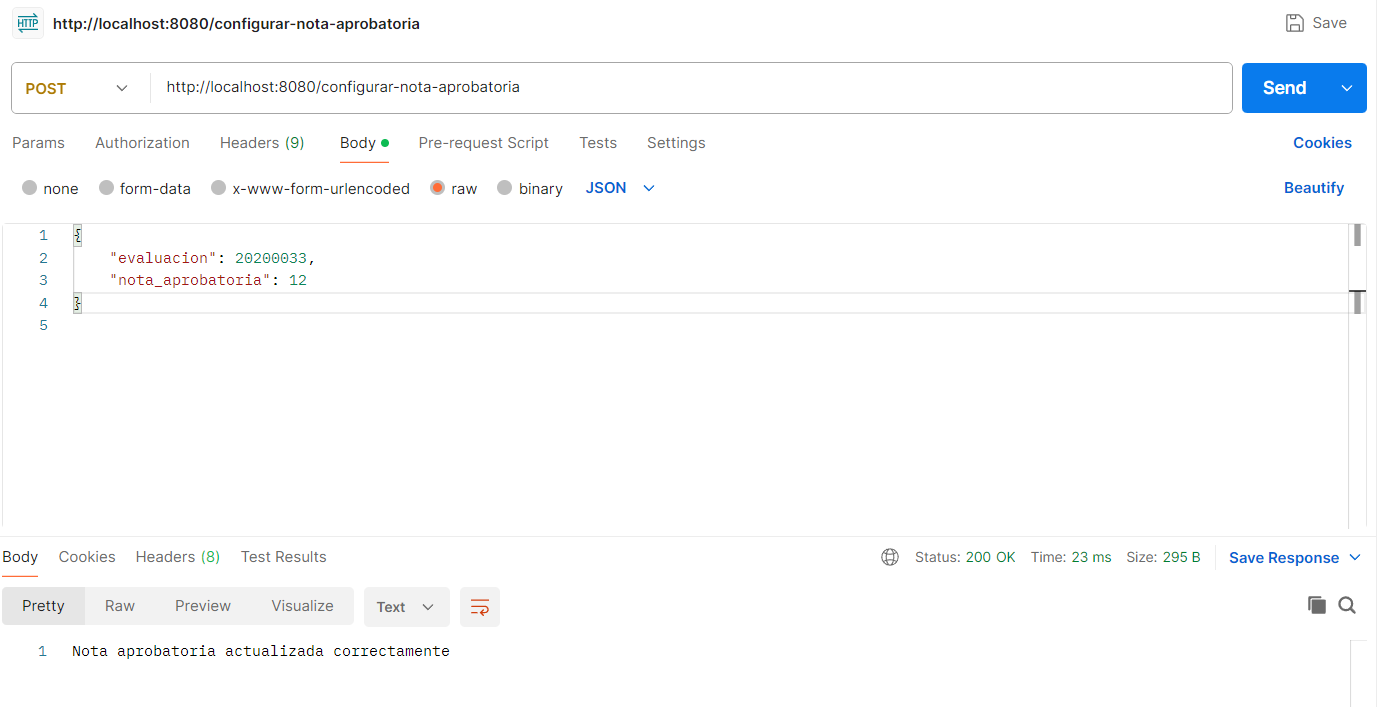
{

"evaluacion": 20200033,

"nota\_aprobatoria": 12

}

**Resultado:**

****

1. **Obtener respuestas de un alumno:** Obtiene todas las respuestas de un alumno para una evaluación, junto con su calificación, la nota aprobatoria y si ha aprobado o no. Método HTTP POST.

**Endpoint**: /obtener-respuestas-alumno

**Parámetros**:

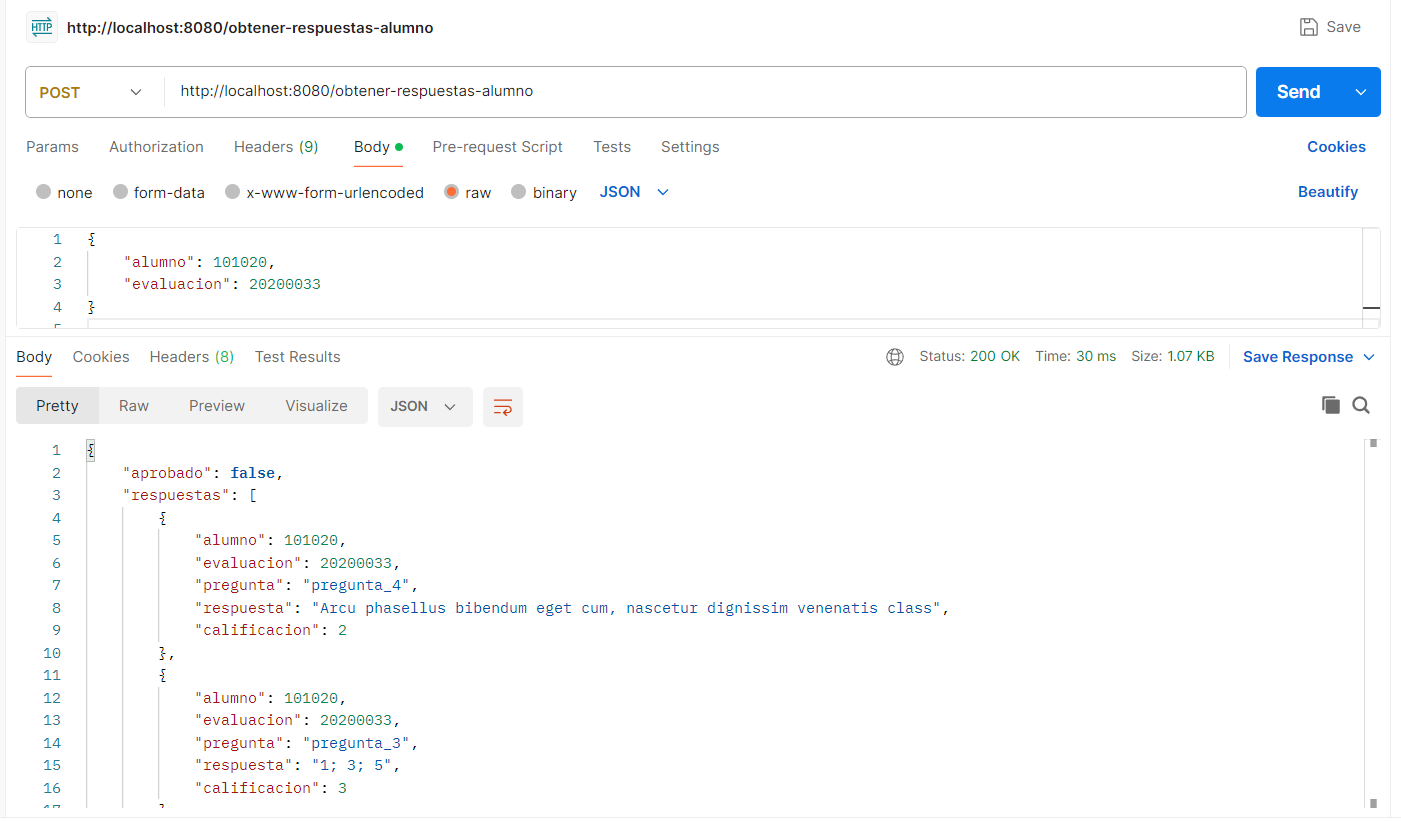
* **Body (JSON)**:

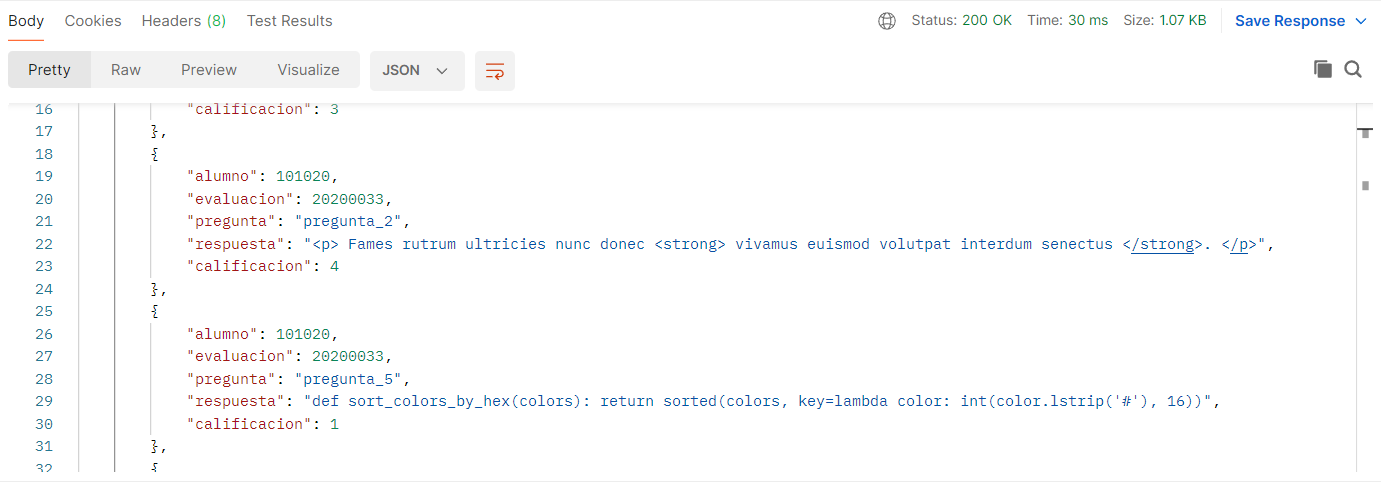
{

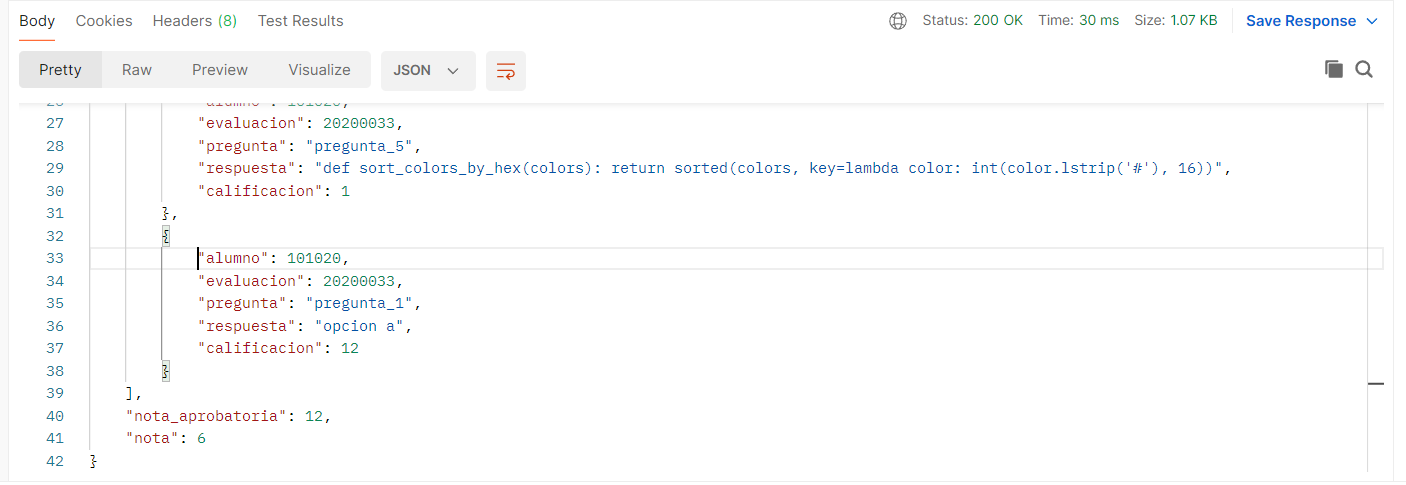
"alumno": 101020,

"evaluacion": 20200033

}

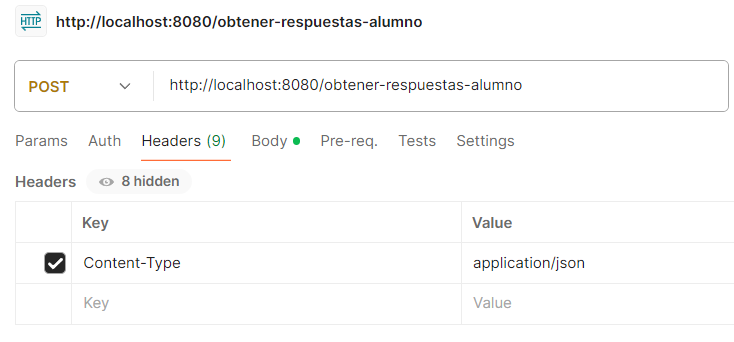
**Respuesta:  
**

****

****

# **Recomendaciones**

* No olvidar colocar el Content-Type y application/json

****

* Solo se usó Body: Raw en formato JSON