Programa Expedition Trainee – UTP

POSTULANTE: MIGUEL ANGEL GARCIA CASIQUE

ÁREA: BACKEND

FECHA: LUNES - 23/09/2024

Tabla de Contenido

1. Casos de Uso 4

A. Caso de Uso: Guardar respuestas de evaluación 4

B. Caso de Uso: Calificar respuestas de evaluación 4

C. Caso de Uso: Configurar Peso de las preguntas 5

D. Caso de Uso: Configurar la Nota Aprobatoria de la evaluación 5

E. Caso de Uso: Obtener promedio de la evaluación 5

2. Diagramas de Proceso 6

F. Diagrama de Proceso: Guardar respuestas de evaluación 6

G. Diagrama de Proceso: Calificar respuestas de evaluación 6

H. Diagrama de Proceso: Configurar Peso de las preguntas 7

I. Diagrama de Proceso: Configurar la Nota Aprobatoria de la evaluación 7

J. Diagrama de Proceso: Obtener promedio de la evaluación 7

3. Diccionario de Datos 8

A. Tabla: alumno 8

B. Tabla: evaluador 8

C. Tabla: evaluación 8

D. Tabla: preguntas 8

E. Tabla: respuestas_alumno 9

F. Tabla: notas 9

G. Tabla: asignacion_evaluador 10

4. Modelo Entidad – Relación 10

5. Librerías 10

6. Componentes 11

A. Controladores 11

B. Servicios 11

C. Repositorios 11

D. Modelos 11

7. Base de Datos 12

8. Catálogo de Servicios Web 16

A. Guardar respuestas del alumno 16

B. Modificar el peso de una pregunta 17

C. Modificar la nota aprobatoria de la evaluación 19

D. Calificar la respuesta de un alumno 19

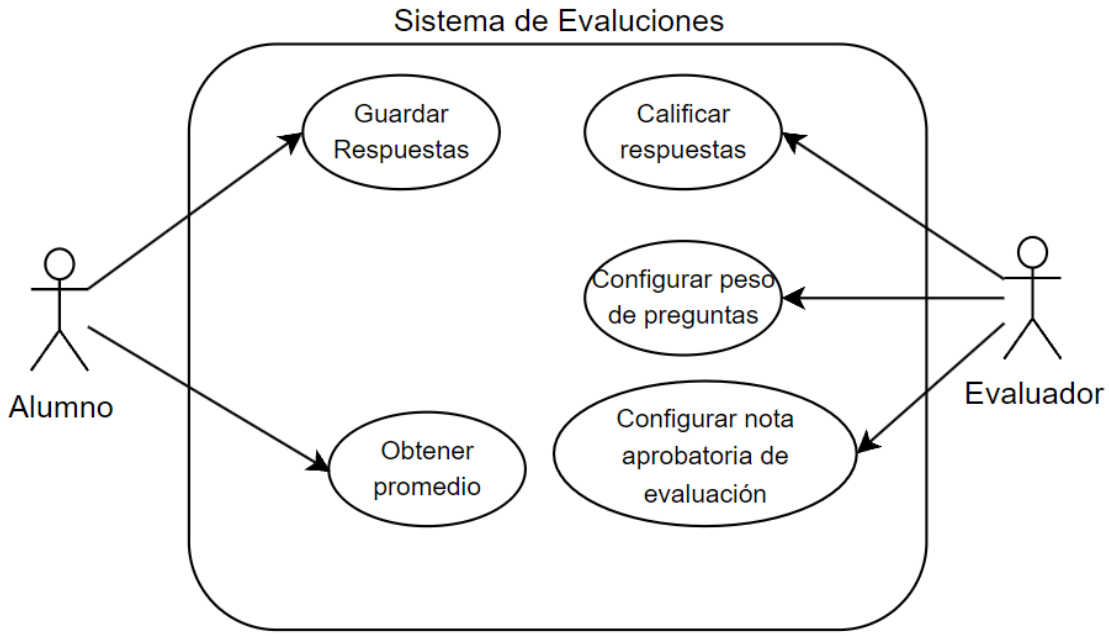
E. Cerrar la evaluación 21

F. Revisar respuesta por pregunta 22

G. Calcular el promedio total.....23
9. Consideraciones Finales.....24

1. Casos de Uso

Utilizando casos de uso, podemos identificar requisitos funcionales del sistema y expresarlos desde el punto de vista del usuario. Antes de describir cada caso de uso, primero se mostrará el diagrama de casos de uso para el sistema desarrollado.



A. Caso de Uso: Guardar respuestas de evaluación

Actor Principal	Alumno
Descripción	El alumno completa su evaluación enviando sus respuestas para las preguntas que contestó.
Precondición	El alumno debe estar registrado y la evaluación debe estar activa
Secuencia Normal	1. El alumno ingresa a la evaluación. 2. El sistema le muestra las preguntas de la evaluación. 3. El alumno responde las preguntas. 4. El sistema hace un seguimiento de las respuestas dadas por el alumno. 5. El alumno envía sus respuestas 6. El sistema almacena las respuestas en la base de datos
Postcondición	Las respuestas son registradas en la base de datos
Excepciones	• Si el usuario no está registrado. • Si la evaluación no existe • Si la evaluación no tiene preguntas asignadas.
Comentarios	Es posible que el alumno no responda todas las preguntas.

B. Caso de Uso: Calificar respuestas de evaluación

Actor Principal	Evaluador
Descripción	El evaluador asignado a una evaluación puede calificar las respuestas del alumno a las preguntas. Solo evaluadores autorizados pueden realizar esta acción.
Precondición	El evaluador debe estar asignado a la evaluación y la evaluación debe estar activa para ser calificada.
Secuencia Normal	1. El evaluador ingresa a la evaluación. 2. El sistema muestra la lista de alumnos que respondieron la evaluación.

	3. El evaluador selecciona un alumno. 4. El sistema muestra las preguntas con la respuesta dada por el alumno seleccionado. 5. El evaluador asigna una calificación a la respuesta. 6. El sistema valida que el evaluador está autorizado para calificar. 7. El evaluador guarda los cambios. 8. El sistema guarda las calificaciones en la base de datos
Postcondición	La calificación se almacena correctamente y se asocia con la respuesta del alumno.
Excepciones	• Si el usuario no está registrado. • Si la evaluación no existe • Si el evaluador no está asignado para calificar.
Comentarios	Luego de calificar a todos los alumnos, el evaluador deberá cambiar el estado de la evaluación a cerrado.

C. Caso de Uso: Configurar Peso de las preguntas

Actor Principal	Evaluador
Descripción	El evaluador puede configurar el peso que tendrá una pregunta en relación a todas las preguntas establecidas en la evaluación.
Precondición	La evaluación debe estar creada y las preguntas asignadas
Secuencia Normal	1. El evaluador ingresa a la evaluación. 2. El sistema muestra la lista de preguntas asignadas. 3. El evaluador selecciona una pregunta y establece su peso. 4. El sistema guarda los cambios.
Postcondición	El peso de la pregunta se actualiza en la base de datos y será considerado en el cálculo de la calificación final.
Excepciones	• Si la evaluación no existe • Si la evaluación no tiene preguntas asignadas.
Comentarios	-

D. Caso de Uso: Configurar la Nota Aprobatoria de la evaluación

Actor Principal	Evaluador
Descripción	El administrador establece o modifica la nota mínima necesaria para aprobar la evaluación. Esto permite definir el criterio para aprobar o reprobar la evaluación.
Precondición	La evaluación debe estar creada.
Secuencia Normal	1. El administrador selecciona una evaluación. 2. El sistema permite ingresar o modificar la nota aprobatoria. 3. El administrador guarda la nota aprobatoria.
Postcondición	La nota aprobatoria se guarda en la base de datos y será utilizada para determinar si un alumno aprobó o no la evaluación.
Excepciones	• Si la evaluación no existe • Si el valor de la nota es inválido.
Comentarios	-

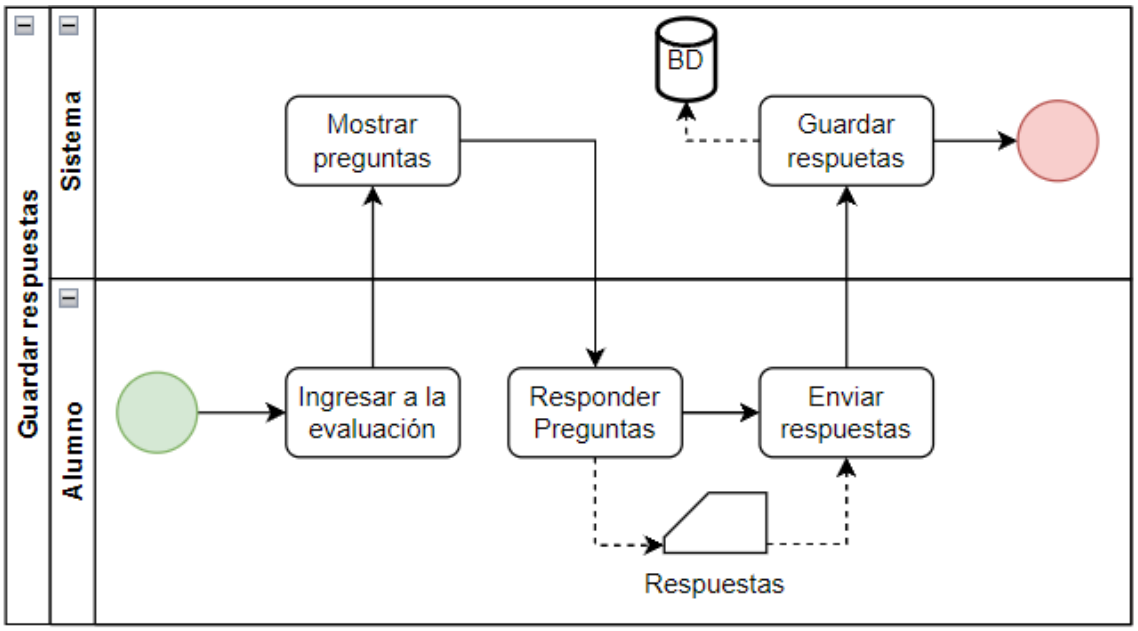
E. Caso de Uso: Obtener promedio de la evaluación

Actor Principal	Alumno
Descripción	El sistema obtiene las respuestas del alumno y muestra las calificaciones de cada pregunta junto con el promedio final de la evaluación, indicando si el alumno ha aprobado o no.

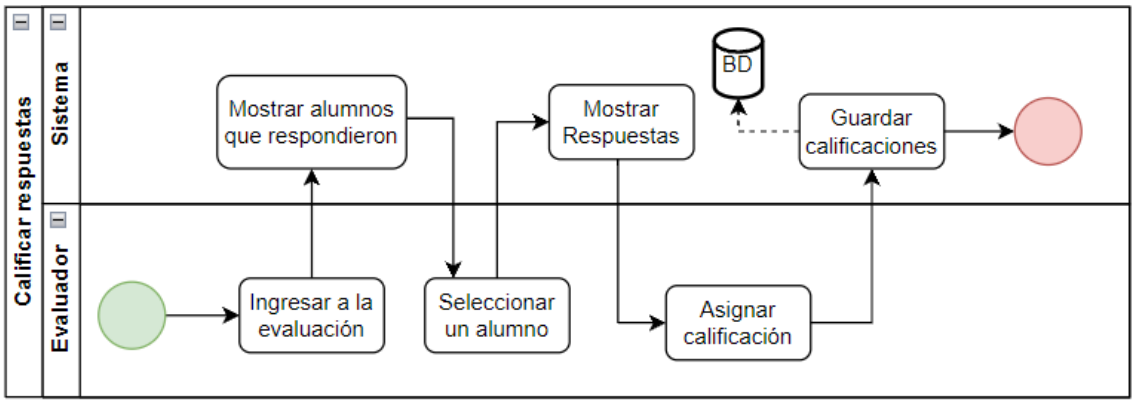
Precondición	El alumno debe haber completado la evaluación y las calificaciones deben haber sido asignadas a las preguntas.
Secuencia Normal	1. El alumno ingresa a la evaluación. 2. El sistema le muestra todas las respuestas calificadas del alumno y su promedio final. 3. El alumno ve sus resultados
Postcondición	El alumno ve el resultado y el promedio total de la evaluación.
Excepciones	• Si las preguntas no han sido calificadas
Comentarios	-

2. Diagramas de Proceso

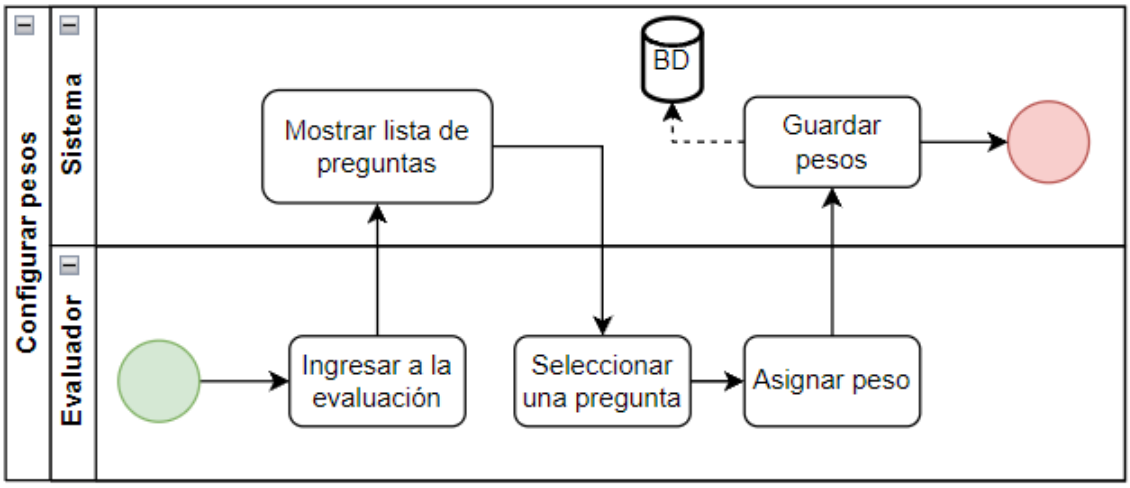
F. Diagrama de Proceso: Guardar respuestas de evaluación



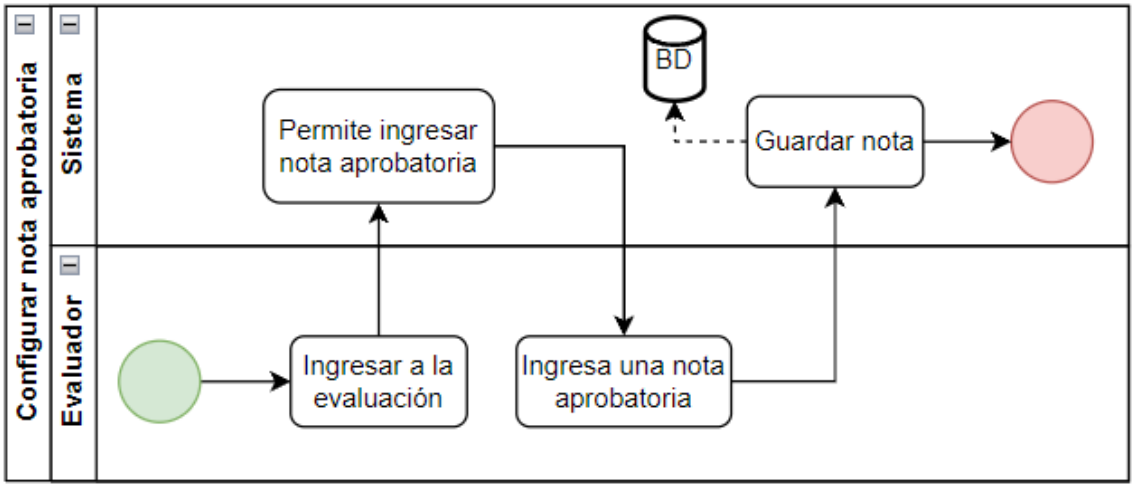
G. Diagrama de Proceso: Calificar respuestas de evaluación



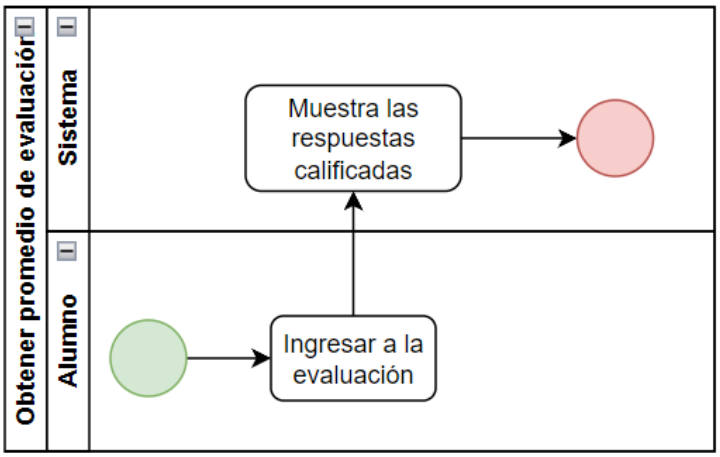
H. Diagrama de Proceso: Configurar Peso de las preguntas



I. Diagrama de Proceso: Configurar la Nota Aprobatoria de la evaluación



J. Diagrama de Proceso: Obtener promedio de la evaluación



3. Diccionario de Datos

A. Tabla: alumno

Nombre del Campo	Tipo de Dato	Descripción	Restricciones
id_alumno	INTEGER	Identificador único del alumno. Generado automáticamente con secuencia.	PRIMARY KEY, AUTOINCREMENT (SERIAL)
nombre	VARCHAR(100)	Nombre completo del alumno.	NOT NULL

B. Tabla: evaluador

Nombre del Campo	Tipo de Dato	Descripción	Restricciones
id_evaluador	INTEGER	Identificador único del evaluador. Generado automáticamente con secuencia.	PRIMARY KEY, AUTOINCREMENT (SERIAL)
nombre	VARCHAR(100)	Nombre completo del evaluador.	NOT NULL

C. Tabla: evaluación

Nombre del Campo	Tipo de Dato	Descripción	Restricciones
id_evaluacion	INTEGER	Identificador único de la evaluación. Generado automáticamente con secuencia.	PRIMARY KEY, AUTOINCREMENT (SERIAL)
curso	VARCHAR(100)	Nombre del curso asociado a la evaluación.	NULLABLE
nota_aprobatoria	DECIMAL(5, 2)	Nota mínima para aprobar la evaluación.	NULLABLE
estado	VARCHAR(50)	Estado de la evaluación (por ejemplo: activo, cerrado).	NULLABLE

D. Tabla: preguntas

Nombre del Campo	Tipo de Dato	Descripción	Restricciones
id_pregunta	INTEGER	Identificador único de la pregunta. Generado automáticamente con secuencia.	PRIMARY KEY, AUTOINCREMENT (SERIAL)
enunciado	TEXT	Texto de la pregunta.	NOT NULL
peso	DECIMAL(5, 2)	Peso de la pregunta en la evaluación.	NULLABLE
orden	INT	Orden de aparición de la pregunta en la evaluación.	NULLABLE

id_evaluacion	INT	Identificador de la evaluación asociada a la pregunta.	FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion)

E. Tabla: respuestas_alumno

Nombre del Campo	Tipo de Dato	Descripción	Restricciones
id_respuesta	INTEGER	Identificador único de la respuesta. Generado automáticamente con secuencia.	PRIMARY KEY, AUTOINCREMENT (SERIAL)
respuesta	TEXT	Texto de la respuesta proporcionada por el alumno.	NOT NULL
calificacion	DECIMAL(5, 2)	Calificación asignada a la respuesta.	NULLABLE
id_alumno	INT	Identificador del alumno que respondió.	FOREIGN KEY (id_alumno) REFERENCES alumno(id_alumno)
id_evaluacion	INT	Identificador de la evaluación.	FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion)
id_pregunta	INT	Identificador de la pregunta asociada a la respuesta.	FOREIGN KEY (id_pregunta) REFERENCES preguntas(id_pregunta)

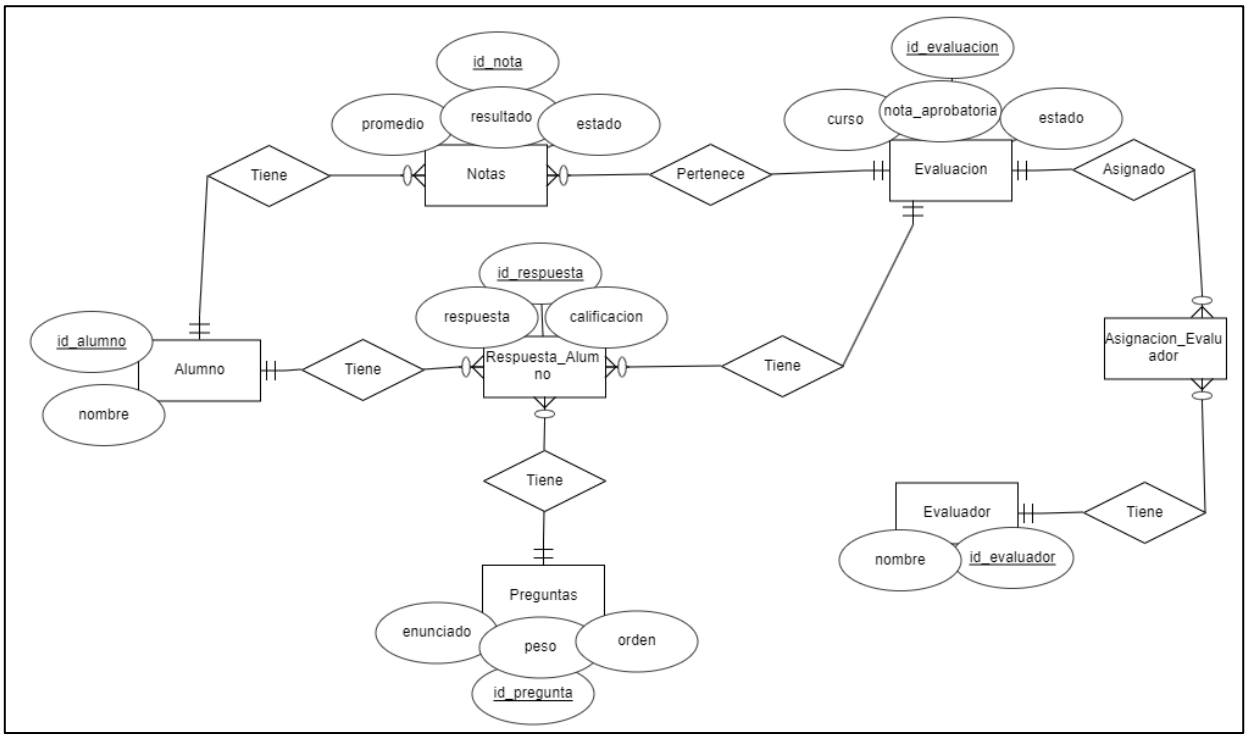
F. Tabla: notas

Nombre del Campo	Tipo de Dato	Descripción	Restricciones
id_nota	INTEGER	Identificador único de la nota. Generado automáticamente con secuencia.	PRIMARY KEY, AUTOINCREMENT (SERIAL)
promedio	DECIMAL(5, 2)	Promedio final de la evaluación del alumno.	NULLABLE
resultado	VARCHAR(50)	Resultado de la evaluación (aprobado/desaprobado).	NULLABLE
estado	VARCHAR(50)	Estado de la nota (activo/cerrado).	NULLABLE
id_alumno	INT	Identificador del alumno.	FOREIGN KEY (id_alumno) REFERENCES alumno(id_alumno)
id_evaluacion	INT	Identificador de la evaluación.	FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion)

G. Tabla: asignacion_evaluador

Nombre del Campo	Tipo de Dato	Descripción	Restricciones
id_asignacion	INTEGER	Identificador único de la asignación. Generado automáticamente con secuencia.	PRIMARY KEY, AUTOINCREMENT (SERIAL)
id_evaluacion	INT	Identificador de la evaluación asignada al evaluador.	FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion)
id_evaluador	INT	Identificador del evaluador.	FOREIGN KEY (id_evaluador) REFERENCES evaluador(id_evaluador)

4. Modelo Entidad – Relación



5. Librerías

Estas están definidas en el archivo pom.xml del proyecto.

Librería	Descripción
Spring Boot Starter Data JPA	Proporciona soporte para Spring Data JPA, facilitando la persistencia de datos en bases de datos relacionales mediante JPA y repositorios.
Spring Boot Starter Web	Incluye dependencias para desarrollar aplicaciones web con Spring MVC y exponer servicios RESTful.
Spring Boot Devtools	Herramientas de desarrollo para acelerar el ciclo de desarrollo en Spring Boot con recarga automática de la aplicación.
Postgresql	Driver JDBC para conectar tu aplicación con bases de datos PostgreSQL.
Lombok	Proporciona anotaciones para generar automáticamente getters, setters y otros métodos estándar, reduciendo código repetitivo en Java.

6. Componentes

Se utilizó la siguiente estructura para el proyecto:

A. Controladores

Con los controladores manejamos las solicitudes HTTP entrantes y se devuelven las respuestas HTTP. Se tienen los siguientes endpoints:

- /evaluaciones/modificar-notaAprobatoria: Usada para establecer la nota mínima aprobatoria de la evaluación
- /evaluaciones/cerrar: Usada para cambiar el estado de la evaluación de "Activo" a "Cerrado", para evitar cambios en las calificaciones.
- /notas/calcular: Usada para calcular el promedio de la evaluación para un determinado alumno.
- /preguntas/modificar-peso: Usada para establecer el peso de una pregunta.
- /respuestas/guardar: Usada para guardar las respuestas de los alumnos.
- /respuestas/…/alumno: Usada para obtener la respuesta de una evaluación, de un determinado alumno y de una determinada pregunta
- /respuestas/…/pregunta: Usada para calificar la respuesta de una determinada pregunta, de un alumno en una evaluación.

B. Servicios

Los servicios encapsulan la lógica de negocio de la aplicación. Se encargan de procesar las solicitudes de los controladores y manejar las reglas de negocio antes de interactuar con los repositorios o devolver respuestas.

En esta parte se maneja todo lo relacionado con la evaluación, las respuestas y las notas finales de los alumnos.

C. Repositorios

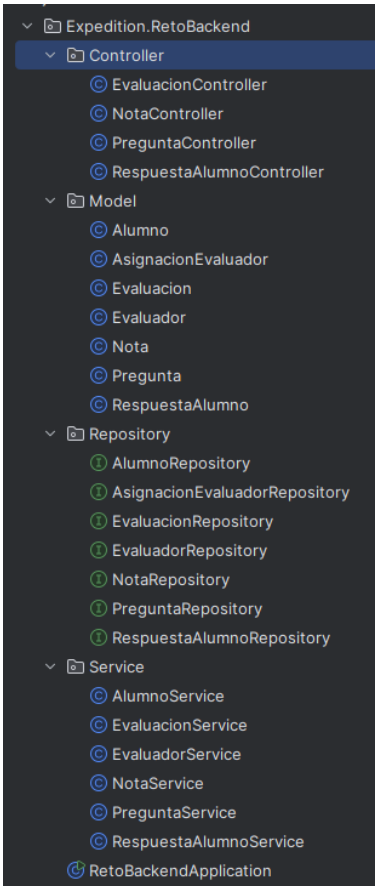
Los repositorios son los componentes encargados de interactuar directamente con la base de datos. Se encargan de realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y otras consultas personalizadas sobre las entidades.

Todos los modelos poseen un repositorio propio.

D. Modelos

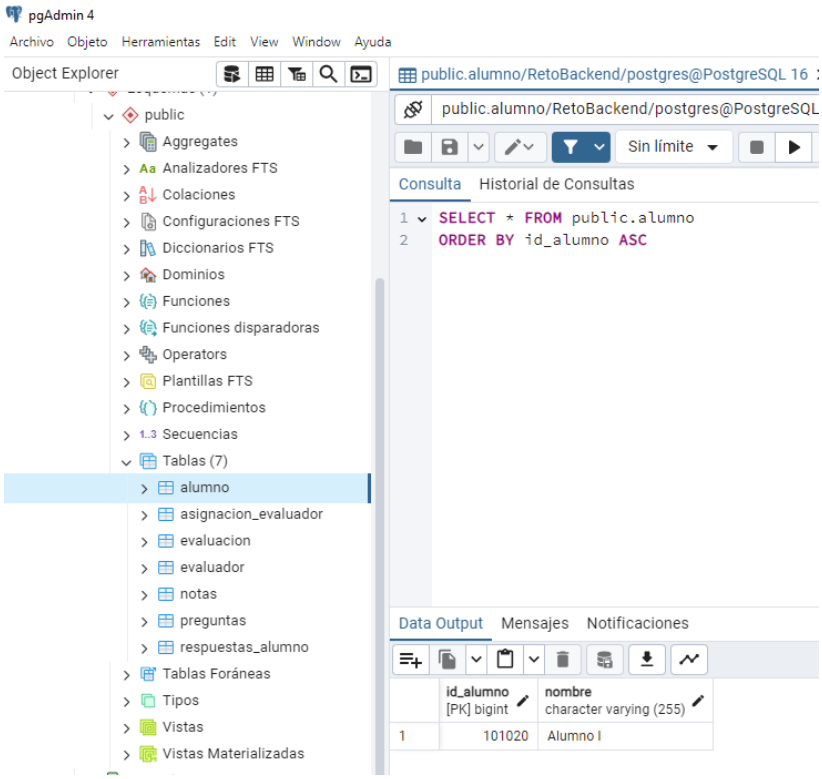
Son clases que representan las tablas en la base de datos. Cada entidad está mapeada a una tabla en la base de datos, y sus atributos corresponden a las columnas de la tabla. Estas entidades son utilizadas por los repositorios para realizar operaciones de persistencia.

A continuación, se muestra una imagen con todos los componentes mencionados.



7. Base de Datos

Para la base de datos se utilizó **PostgreSQL**, en específico la versión 16.



Se utilizó el siguiente script para crear base de datos inicial. (Ejecutar cada línea por separado)

```
DROP DATABASE IF EXISTS retobackend;
CREATE DATABASE retobackend;
```

Una vez creada la BD, usamos la misma para crear las tablas e insertar algunos datos para las pruebas.

```
DROP TABLE IF EXISTS asignacion_evaluador;
DROP TABLE IF EXISTS notas;
DROP TABLE IF EXISTS respuestas_alumno;
DROP TABLE IF EXISTS preguntas;
DROP TABLE IF EXISTS evaluacion;
DROP TABLE IF EXISTS evaluador;
DROP TABLE IF EXISTS alumno;
CREATE TABLE alumno (
  id_alumno SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

CREATE TABLE evaluador (
  id_evaluador SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

CREATE TABLE evaluacion (
  id_evaluacion SERIAL PRIMARY KEY,
  curso VARCHAR(100),
  nota_aprobatoria DECIMAL(5, 2),
  estado VARCHAR(50)
);

CREATE TABLE preguntas (
  id_pregunta SERIAL PRIMARY KEY,
  enunciado TEXT NOT NULL,
  peso DECIMAL(5, 2),
  orden INT,
  id_evaluacion INT,
  FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion)
);

CREATE TABLE respuestas_alumno (
  id_respuesta SERIAL PRIMARY KEY,
  respuesta TEXT NOT NULL,
  calificacion DECIMAL(5, 2),
  id_alumno INT,
  id_evaluacion INT,
  id_pregunta INT,
  FOREIGN KEY (id_alumno) REFERENCES alumno(id_alumno),
  FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion),
  FOREIGN KEY (id_pregunta) REFERENCES preguntas(id_pregunta)
```

```
);

CREATE TABLE notas (
   id_nota SERIAL PRIMARY KEY,
   promedio DECIMAL(5, 2),
   resultado VARCHAR(50),
   estado VARCHAR(50),
   id_alumno INT,
   id_evaluacion INT,
   FOREIGN KEY (id_alumno) REFERENCES alumno(id_alumno),
   FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion)
);

CREATE TABLE asignacion_evaluador (
   id_asignacion SERIAL PRIMARY KEY,
   id_evaluacion INT,
   id_evaluador INT,
   FOREIGN KEY (id_evaluacion) REFERENCES evaluacion(id_evaluacion),
   FOREIGN KEY (id_evaluador) REFERENCES evaluador(id_evaluador)
);

--Insertando valores ALUMNO
INSERT INTO alumno(
        id_alumno, nombre)
        VALUES (101020,  'Alumno I');

--Insertando valores EVALUACION
INSERT INTO evaluacion(
        id_evaluacion, curso, nota_aprobatoria, estado)
        VALUES (20200033,  'Curso I', null, 'Activo');

INSERT INTO evaluacion(
        id_evaluacion, curso, nota_aprobatoria, estado)
        VALUES (1, 'Prueba', null, 'Activo');

--Insertando valores PREGUNTAS
INSERT INTO preguntas(
        id_pregunta, enunciado, peso, orden, id_evaluacion)
        VALUES (1, 'Enun 1', null, 1, 20200033);

INSERT INTO preguntas(
        id_pregunta, enunciado, peso, orden, id_evaluacion)
        VALUES (2, 'Enun 2', null, 2, 20200033);

INSERT INTO preguntas(
        id_pregunta, enunciado, peso, orden, id_evaluacion)
        VALUES (3, 'Enun 3', null, 3, 20200033);

INSERT INTO preguntas(
        id_pregunta, enunciado, peso, orden, id_evaluacion)
        VALUES (4, 'Enun 4', null, 4, 20200033);
```

```
INSERT INTO preguntas(
	id_pregunta, enunciado, peso, orden, id_evaluacion)
	VALUES (5, 'Enun 5', null, 5, 20200033);

--Insertando valores EVALUADORES
INSERT INTO evaluador(
	id_evaluador, nombre)
	VALUES (1, 'Eval I');

INSERT INTO evaluador(
id_evaluador, nombre)
VALUES (2, 'Eval II');


--Asignando evaluadores a las evaluaciones
INSERT INTO asignacion_evaluador(
	id_asignacion, id_evaluacion, id_evaluador)
	VALUES (1, 1, 1);

INSERT INTO asignacion_evaluador(
	id_asignacion, id_evaluacion, id_evaluador)
	VALUES (2, 20200033, 2);
```

De igual manera, se estará colocando un Script.txt en la carpeta Resources del sistema.

Ya creado la BD y sus tablas debemos configurar en nuestra app el puerto, la BD, el usuario, etc. Esto lo realizamos en el "application.properties":

```
spring.application.name=RetoBackend
spring.datasource.url=jdbc:postgresql://localhost:5432/retobackend
spring.datasource.username=postgres
spring.datasource.password=(Cologue su contraseña)
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

8. Catálogo de Servicios Web

A continuación, se mostrará el funcionamiento de cada endpoint mencionado con anterioridad, especificando los métodos de petición HTTP que usan y los parámetros necesarios. Seguiremos el orden en el que sucederían las cosas, empezando con el registro de las respuestas del alumno y terminando con el promedio de la evaluación. Utilizaremos **postman** para las pruebas.

Tener en cuenta que la ruta puede variar dependiendo del puerto en la que se ejecute.

A. Guardar respuestas del alumno

Aquí insertamos las respuestas ingresadas por un alumno a las diversas preguntas de una evaluación.

Método: POST

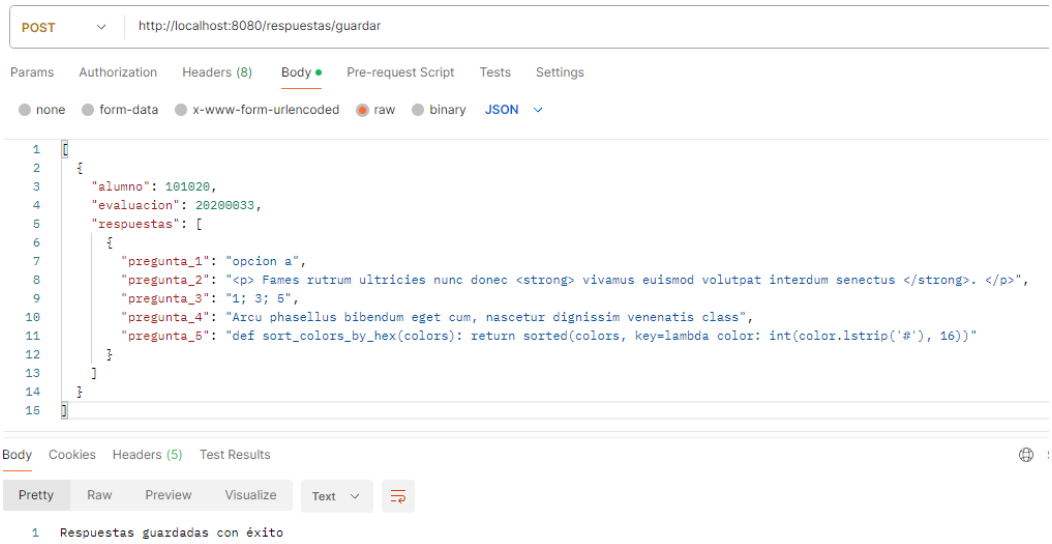
Ruta Completa: http://localhost:8080/respuestas/guardar

Parámetros:

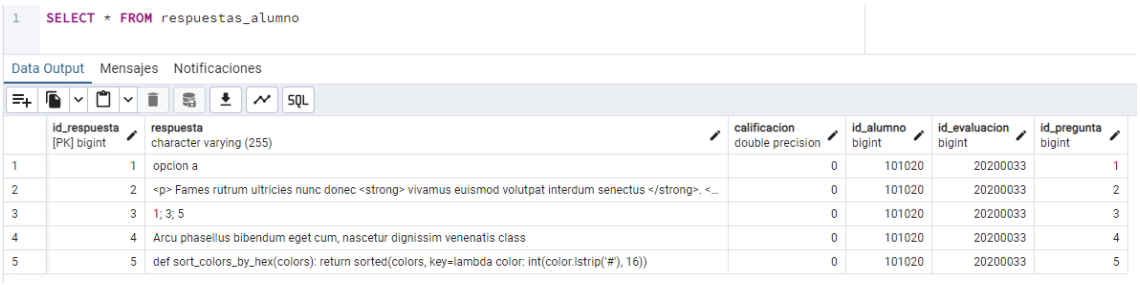
- **Body (raw->JSON):**

```
[
 {
  "alumno": 101020,
  "evaluacion": 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:



En la BD:



```
SELECT * FROM respuestas_alumno
```

Data Output | Mensajes | Notificaciones

	id_respuesta [PK] bigint	respuesta character varying (255)	calificacion double precision	id_alumno bigint	id_evaluacion bigint	id_pregunta bigint
1	1	opcion a	0	101020	20200033	1
2	2	<p> Fames rutrum ultricies nunc donec vivamus euismod volutpat interdum senectus . <...	0	101020	20200033	2
3	3	1; 3; 5	0	101020	20200033	3
4	4	Arcu phasellus bibendum eget cum, nascetur dignissim venenatis class	0	101020	20200033	4
5	5	def sort_colors_by_hex(colors): return sorted(colors, key=lambda color: int(color.lstrip('#'), 16))	0	101020	20200033	5

B. Modificar el peso de una pregunta

Establece el peso de una pregunta en base al total de la evaluación. Si bien esto debería ser lo primero que el evaluador debería modificar, no influye mucho si aún no se busca colgar notas en el sistema.

Método: POST

Ruta Completa:

http://localhost:8080/preguntas/ modificar-peso/{idEvaluacion}/Preg/{nroPregunta}

Parámetros:

- **Params:**

peso:2

Resultado:

POST http://localhost:8080/preguntas/modificar-peso/20200033/Preg/1?peso=2

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	Key
☑	peso

ody Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1  Se actulizo el peso de la pregunta correctamente.
```

En la BD:

```
1  SELECT * FROM preguntas order by orden
```

Data Output Mensajes Notificaciones

	id_pregunta [PK] bigint	enunciado character varying (255)	peso double precision	orden integer	id_evaluacion bigint
1	1	Enun 1	2	1	20200033
2	2	Enun 2	[null]	2	20200033
3	3	Enun 3	[null]	3	20200033
4	4	Enun 4	[null]	4	20200033
5	5	Enun 5	[null]	5	20200033

Supongamos que lo hicimos para todos y queda de la siguiente manera:

```
1  SELECT * FROM preguntas order by orden
```

Data Output Mensajes Notificaciones

	id_pregunta [PK] bigint	enunciado character varying (255)	peso double precision	orden integer	id_evaluacion bigint
1	1	Enun 1	2	1	20200033
2	2	Enun 2	1	2	20200033
3	3	Enun 3	2	3	20200033
4	4	Enun 4	3	4	20200033
5	5	Enun 5	1	5	20200033

C. Modificar la nota aprobatoria de la evaluación

Establece la nota mínima aprobatoria de una evaluación. Si bien esto debería ser lo primero que el evaluador debería modificar, no influye mucho si aún no se busca colgar notas en el sistema.

Método: POST

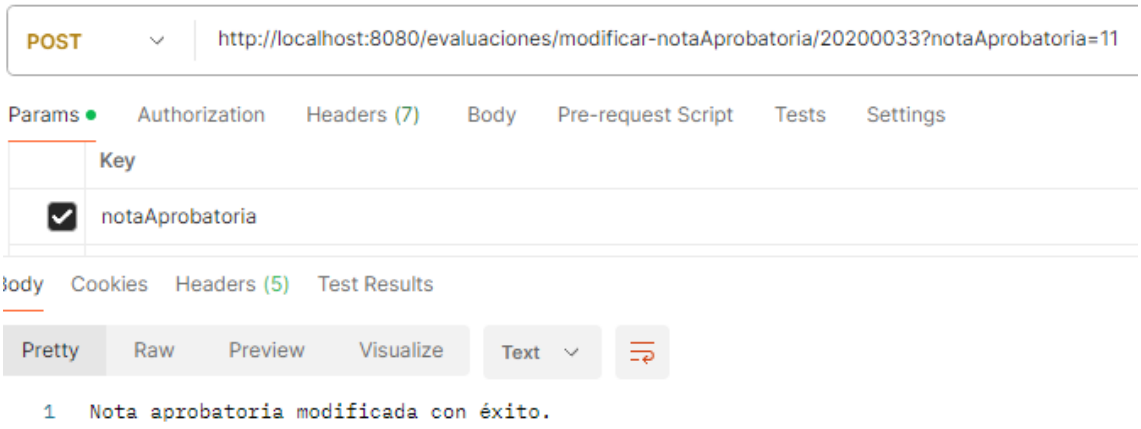
Ruta Completa:

http://localhost:8080/ evaluaciones/ modificar-notaAprobatoria/{idEvaluacion}

Parámetros:

- **Params:**
  notaAprobatoria:11

Resultado:



En la BD:

	id_evaluacion [PK] bigint	curso character varying (255)	nota_aprobatoria double precision	estado character varying (255)
1	1	Prueba	[null]	Activo
2	20200033	Curso I	11	Activo

D. Calificar la respuesta de un alumno

Utilizando el identificador del alumno, de la evaluación y el número de pregunta, el evaluador podrá calificar la evaluación si es que tiene el permiso de hacerlo.

Método: POST

Ruta Completa:

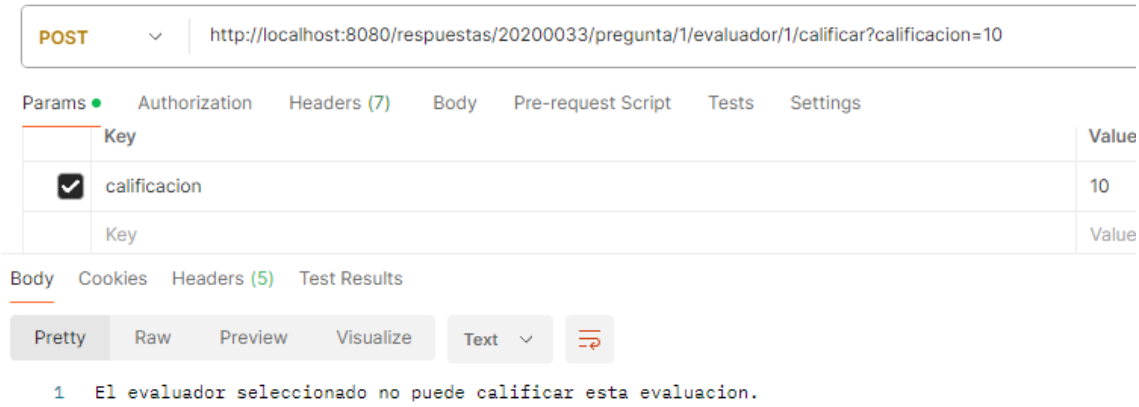
http://localhost:8080/ respuestas
/{idEvaluacion}/pregunta/{nroPregunta}/evaluador/{idEvaluador}/calificar

Parámetros:

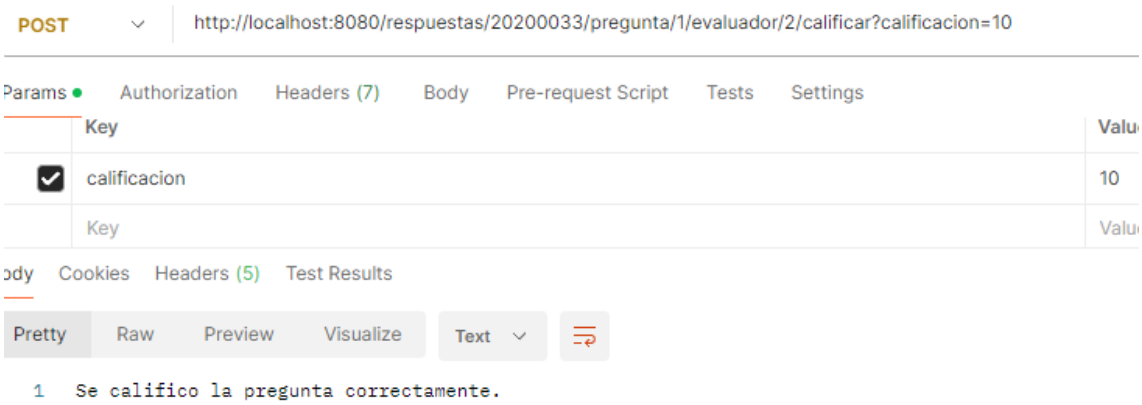
- **Params:**
  calificacion:10

Resultado:

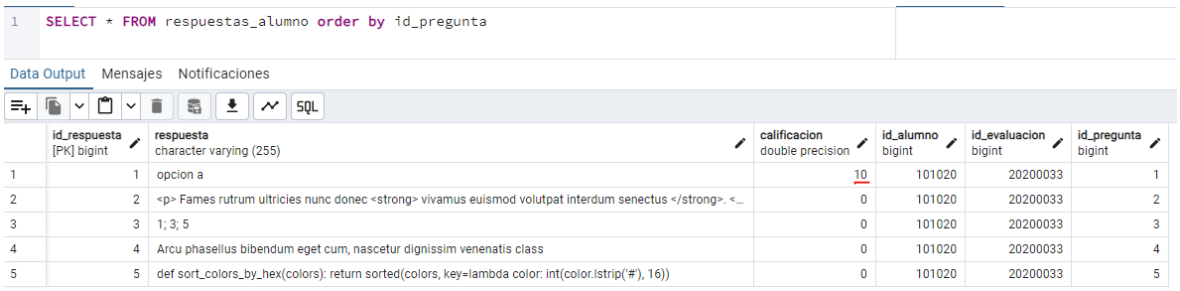
- **Si el evaluador NO tiene permitido calificar**



- **Si el evaluador SI tiene permitido calificar**



En la BD:



	id_respuesta [PK] bigint	respuesta character varying (255)	calificacion double precision	id_alumno bigint	id_evaluacion bigint	id_pregunta bigint
1	1	opcion a	10	101020	20200033	1
2	2	<p> Fames rutrum ultricies nunc donec vivamus euismod volutpat interdum senectus . <...	0	101020	20200033	2
3	3	1; 3; 5	0	101020	20200033	3
4	4	Arcu phasellus bibendum eget cum, nascetur dignissim venenatis class	0	101020	20200033	4
5	5	def sort_colors_by_hex(colors): return sorted(colors, key=lambda color: int(color.lstrip('#'), 16))	0	101020	20200033	5

Supongamos que lo hicimos para todos y queda de la siguiente manera:

```
1  SELECT * FROM respuestas_alumno order by id_pregunta
```

Data Output | Mensajes | Notificaciones

	id_respuesta [PK] bigint	respuesta character varying (255)	calificacion double precision	id_alumno bigint	id_evaluacion bigint	id_pregunta bigint
1	1	opcion a	10	101020	20200033	1
2	2	<p> Fames rutrum ultricies nunc donec vivamus euismod volutpat interdum senectus . <...	5	101020	20200033	2
3	3	1; 3; 5	15	101020	20200033	3
4	4	Arcu phasellus bibendum eget cum, nascetur dignissim venenatis class	20	101020	20200033	4
5	5	def sort_colors_by_hex(colors): return sorted(colors, key=lambda color: int(color.lstrip('#'), 16))	1	101020	20200033	5

E. Cerrar la evaluación

Una vez que el evaluador califica las preguntas de todos los alumnos, entonces este de forma manual debería indicar que las calificaciones ya no se podrán modificar, pasando de un estado activo para la calificación a un estado cerrado o culminado. A esto le sigue la visualización de notas y calificaciones por pregunta por parte del alumno.

Método: POST

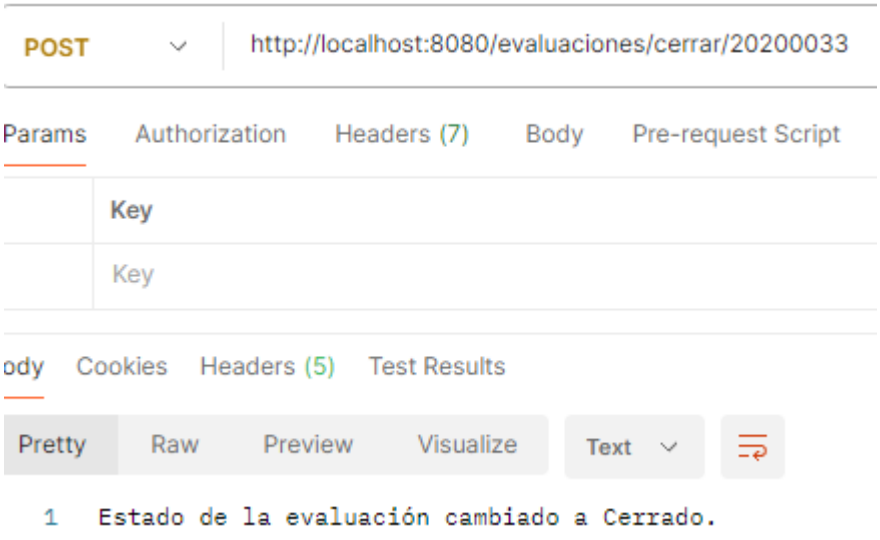
Ruta Completa:

http://localhost:8080/ evaluaciones /cerrar/{idEvaluacion}

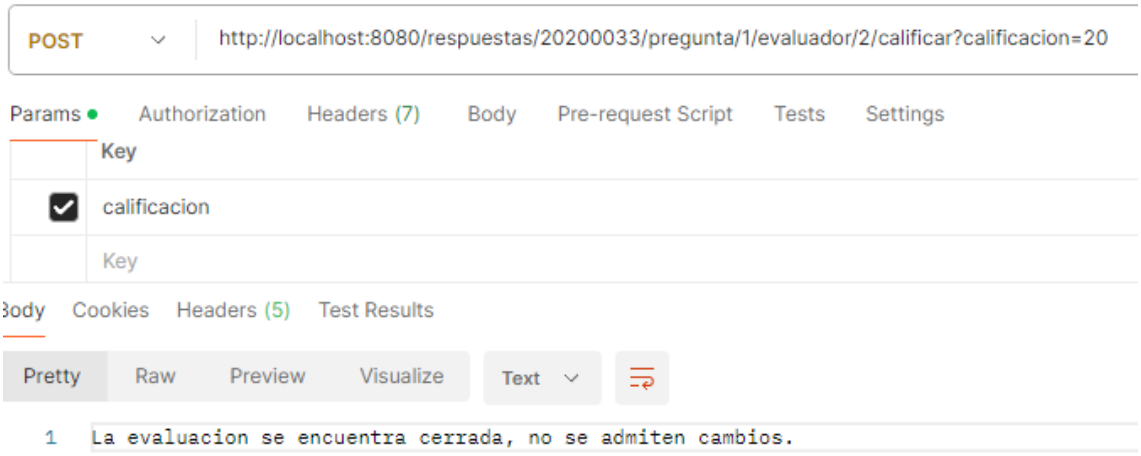
Parámetros:

- **Sin parámetros**

Resultado:



Ahora si se quisiera cambiar la calificación de una pregunta, ya no se podría:



En la BD:

	id_evaluacion [PK] bigint	curso character varying (255)	nota_aprobatoria double precision	estado character varying (255)
1	1	Prueba	[null]	Activo
2	20200033	Curso I	11	Cerrado

F. Revisar respuesta por pregunta

Como medida de visualización. Para conocer que respuesta colocó el alumno en una determinada pregunta de la evaluación.

Método: GET

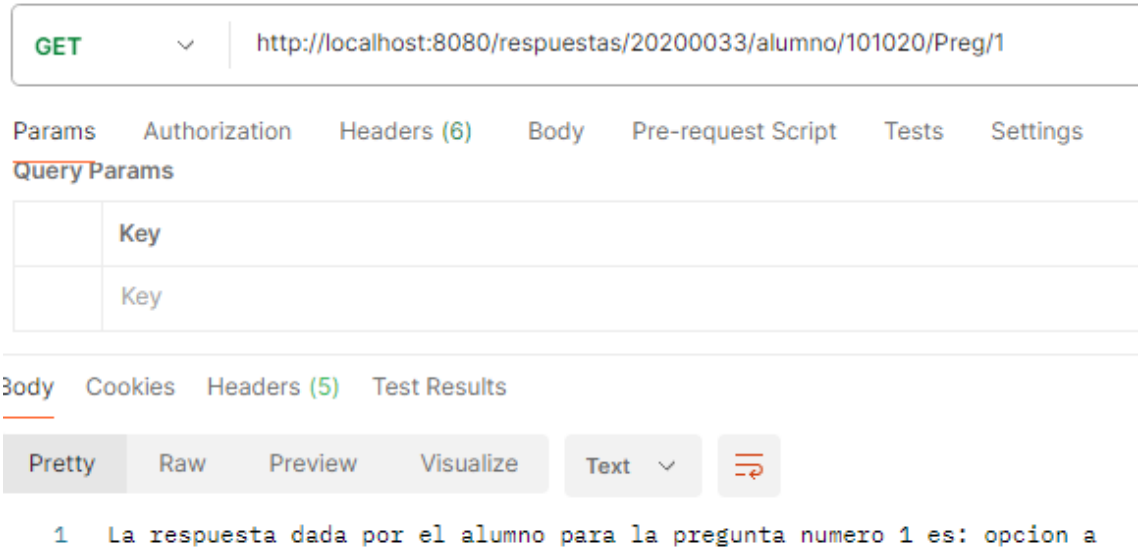
Ruta Completa:

http://localhost:8080/ respuestas/
/{idEvaluacion}/alumno/{idAlumno}/Preg/{nroPregunta}

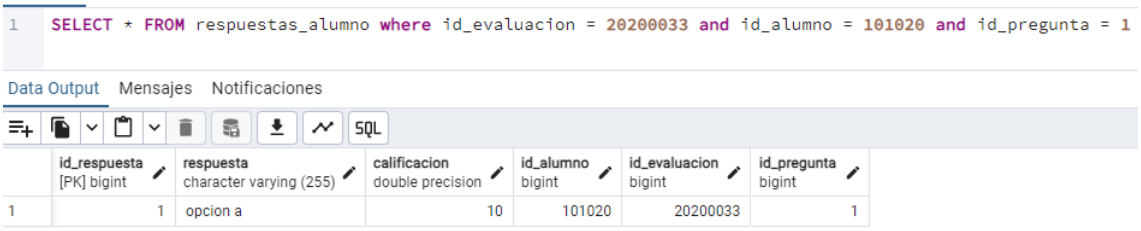
Parámetros:

- **Sin parámetros**

Resultado:



En la BD:



G. Calcular el promedio total

Se utiliza para ponderar en base a los pesos y las calificaciones de las preguntas una nota final de la evaluación. También determina si el alumno aprobó o no en base a la nota aprobatoria de la evaluación.

Método: POST

Ruta Completa:

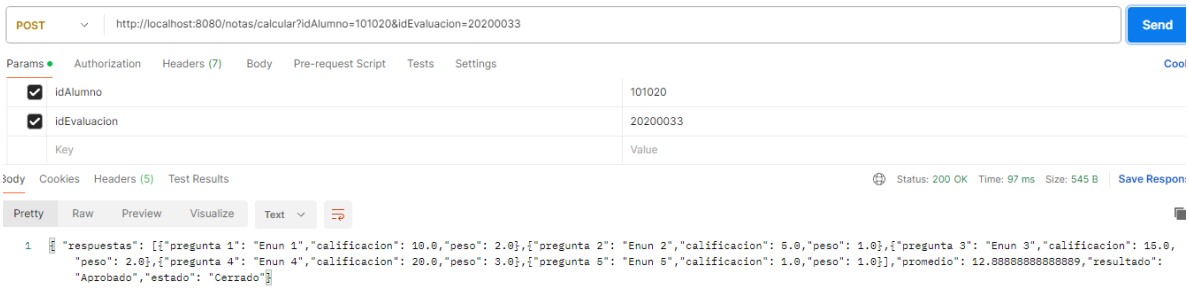
http://localhost:8080/notas/calcular

Parámetros:

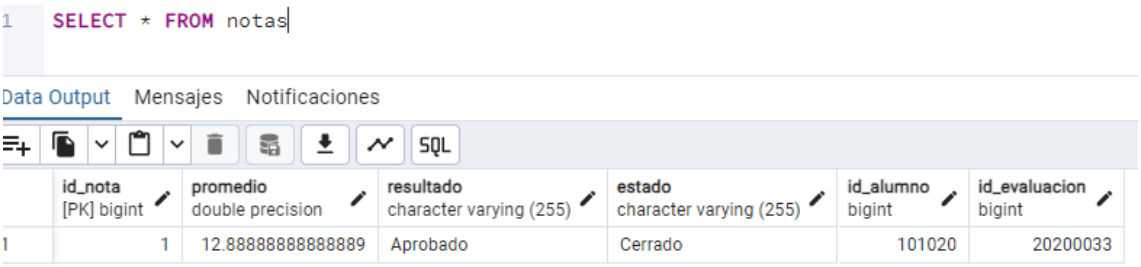
- **Params:**

idAlumno: 101020 idEvaluacion: 20200033

Resultado:



En la BD:



9. Consideraciones Finales

- Se pudieron considerar una mayor densidad de tablas. Como por ejemplo uno cursos, donde cada curso tendría diferentes secciones que podrían compartir un mismo examen. Se pudo colocar una tabla que relacione de muchos a muchos las tablas de preguntas con evaluaciones, ya que una misma pregunta podría estar en evaluaciones pasadas. Para este caso lo dejamos como que una pregunta solo puede aparecer en una sola evaluación.
- Se realizaron algunas labores de validación de existencia de alumnos, evaluaciones y preguntas. Si bien esta puede no ser necesario al momento de evaluar, ya que se esta realizando un seguimiento de todo, para esta app si es necesario.
- Se pudo ahondar más en los casos de uso, como la posibilidad de agregar evaluaciones, agregar preguntas a estas evaluaciones y seguir analizando a profundidad. Pero se dejó lo más significativo para la resolución de este reto.
- Si bien se usaron parámetros en la misma url para guardar datos con POST, es más recomendable usar un json en el body.