Tabla de contenido

1 Casos de Uso 2

1.1 CU-001: Registrar alumno..... 2

1.2 CU-002: Registrar Pago..... 2

1.3 CU-003: Obtener pagos/pagos pendientes 3

2 Diagramas de procesos 3

2.1 Proceso Registrar Alumno..... 4

2.2 Proceso Registrar Pago..... 5

2.3 Proceso obtener todo los pagos y pagos pendientes. 6

3 Diagrama entidad – relación.....6

4 Diccionario de datos.....6

5 Inventario de librerías y componentes 7

5.1 Tecnologías y librerías utilizadas 8

5.2 Componentes..... 8

6 Ejecución del proyecto 10

6.1 Ejecución local 10

6.2 Ejecución en la Nube..... 10

7 Catálogo de servicios web implementados 11

7.1 Registro de alumno..... 11

7.2 Listar alumnos 12

7.3 Registrar Pago..... 12

7.4 Listar Pagos 14

7.5 Listar Pagos Pendientes..... 15

1 Casos de Uso

1.1 CU-001: Registrar alumno

CU-001	Registrar alumno
Descripción	Este caso de uso permite registrar un nuevo alumno en el sistema.
Precondición	El actor debe tener acceso al sistema.
Flujo principal	1. El actor envía una solicitud de registro al sistema con los datos del alumno en formato JSON. 2. El sistema valida los datos recibidos. Verifica que los nombres, apellidos y código no sean nulos. Además, verifica que el código no haya sido usado en otro alumno registrado. En caso de error, el sistema envía una respuesta detallando los errores. 3. En caso de éxito, el sistema devuelve una respuesta de éxito al actor, donde confirma el registro.
Postcondiciones	El alumno se ha registrado con éxito en la base de datos.
Excepciones	Si faltan datos requeridos, o algún dato no es válido, el sistema devuelve una respuesta indicando los errores.

1.2 CU-002: Registrar Pago

CU-002	Registrar Pago
Descripción	Este caso de uso permite registrar un nuevo pago en el sistema.
Precondición	1. El actor debe tener acceso al sistema. 2. El alumno que corresponde el pago debe ser previamente registrado en el sistema.
Flujo principal	1. El actor envía una solicitud de registro al sistema con los datos del pago en una lista de formato JSON. 2. El sistema valida los datos de cada pago. Verifica que la cuota, estado, fecha de vencimiento, monto y alumno sean correctos.

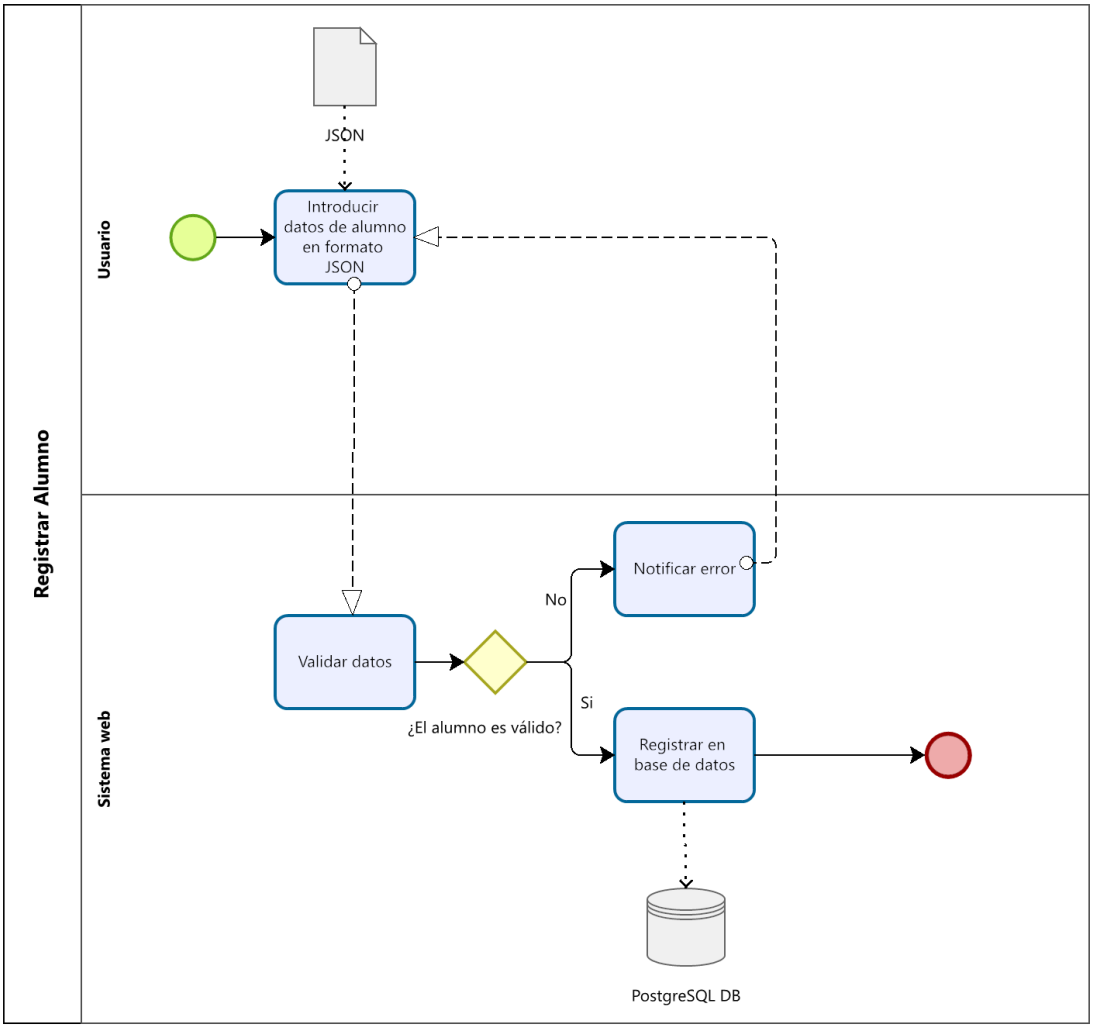
	3. En caso de éxito, el sistema devuelve una respuesta de éxito al actor, donde confirma el registro.
Postcondiciones	El pago se ha registrado con éxito en la base de datos.
Excepciones	Si faltan datos requeridos, o algún dato no es válido, el sistema devuelve una respuesta indicando los errores.

1.3 CU-003: Obtener pagos/pagos pendientes

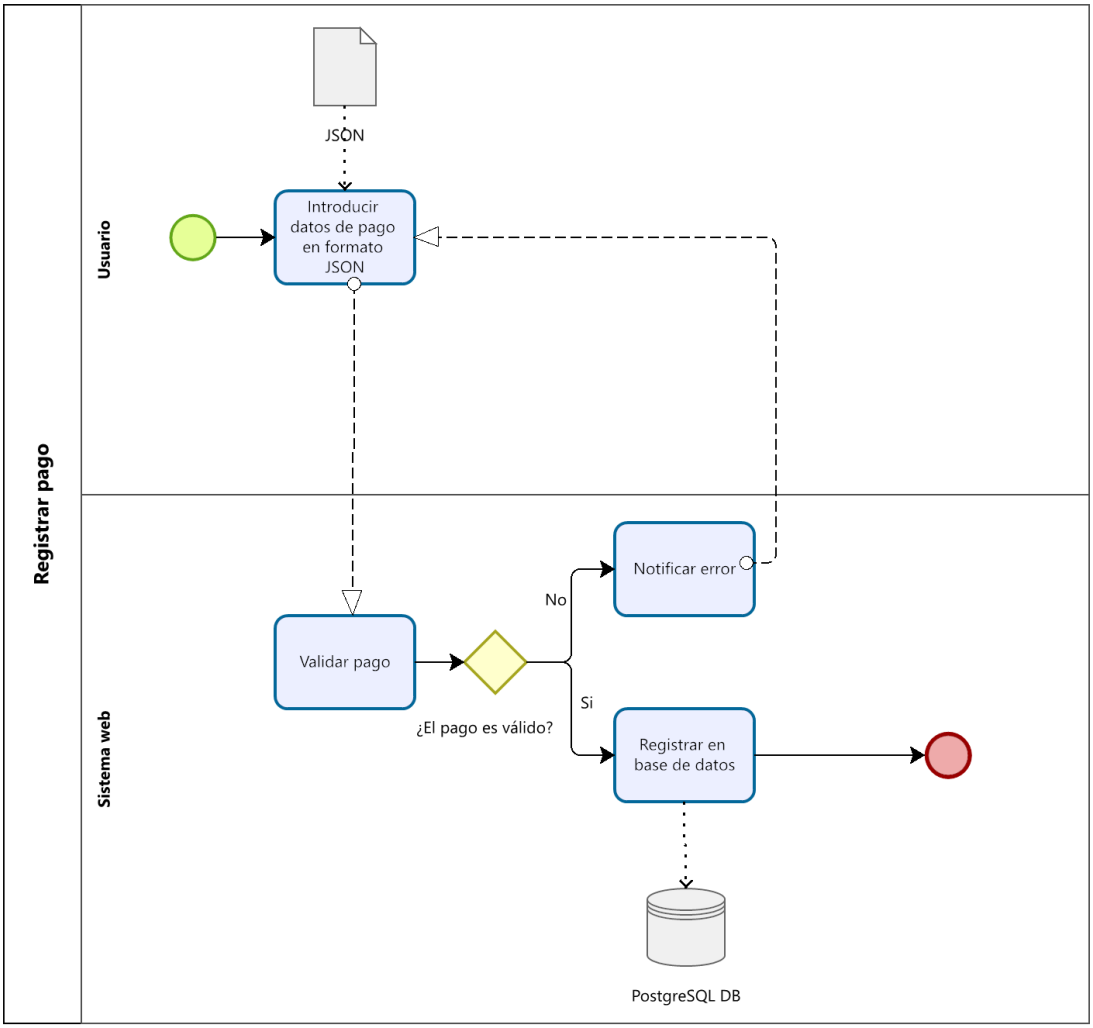
CU-003	Obtener pagos/pagos pendientes
Descripción	Este caso de uso permite obtener los pagos del sistema
Precondición	1. El actor debe tener acceso al sistema.
Flujo principal	1. El actor envía una solicitud al sistema para obtener los pagos pendientes. 2. Se debe añadir la propiedad Accept con valor application/XML en el header para obtener en formato XML solicitado. 3. En caso de éxito, el sistema devuelve una respuesta con todos los pagos de la base de datos
Postcondiciones	Se devuelve una lista de los pagos en la base de datos
Excepciones	Si faltan datos requeridos, o algún dato no es válido, el sistema devuelve una respuesta indicando los errores.

2 Diagramas de procesos

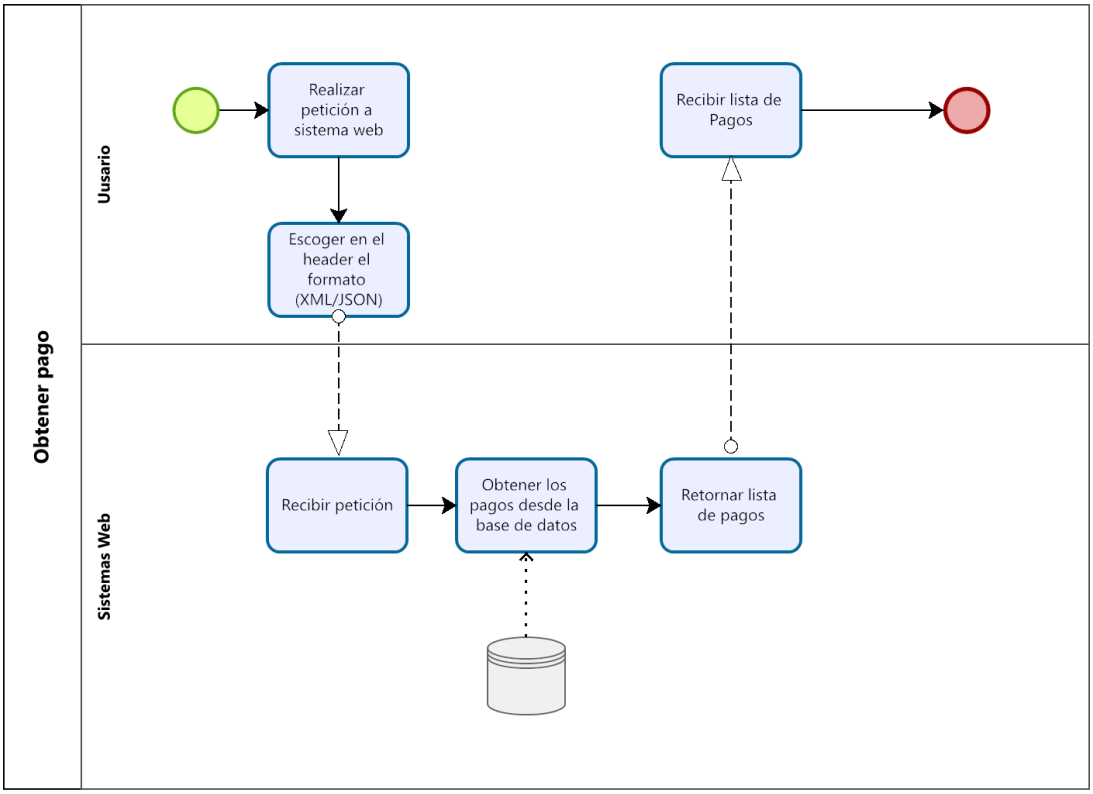
2.1 Proceso Registrar Alumno



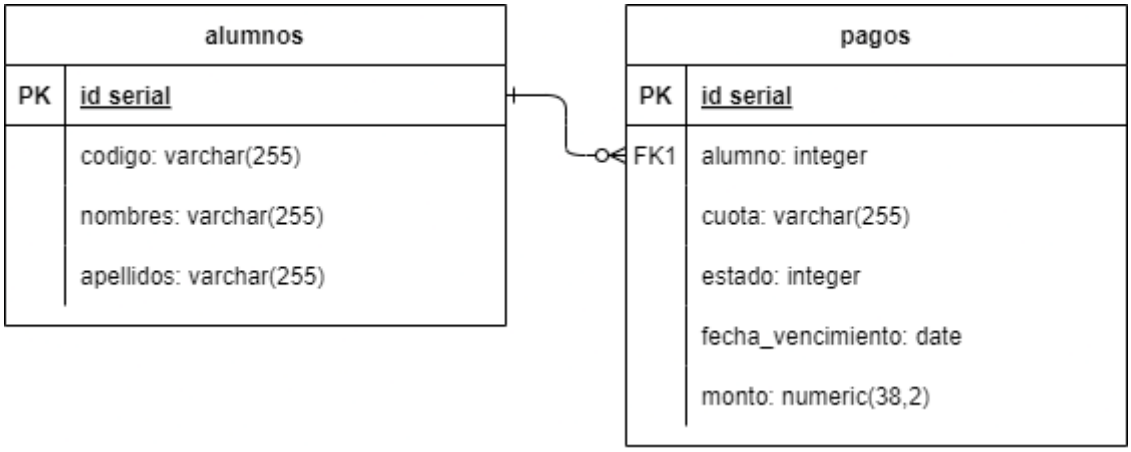
2.2 Proceso Registrar Pago



2.3 Proceso obtener todo los pagos y pagos pendientes.



3 Diagrama entidad – relación



4 Diccionario de datos

Tabla: alumnos

Campo	Descripción	Tipo de dato	Restricciones
id	Identificador único del alumno en la base de datos (PK)	SERIAL	Clave primaria (PK),

			no nulo, autoincremental.
codigo	Código único del alumno	VARCHAR (255)	Único, no nulo.
nombres	Nombres del alumno	VARCHAR (255)	No nulo.
apellidos	Apellidos del alumno	VARCHAR (255)	No nulo.

Detalles

Relaciones: la tabla alumno está relacionada con la entidad pago de 1 a muchos.

Restricciones: El campo Id es la clave primaria y se define como serial para que sea autoincremental. El campo código es único y no puede ser nulo para asegurar que cada alumno tenga un código distinto.

Tabla: pagos

Campo	Descripción	Tipo de dato	Restricciones
id	Identificador único del pago (PK)	SERIAL	Clave primaria (PK), no nulo.
alumno	ID del alumno que le corresponde el pago	INTEGER	Clave foránea (FK), no nulo
cuota	Número de cuota	VARCHAR (255)	No nulo.
monto	Monto en soles correspondiente al pago.	NUMERIC (38,2)	No nulo, mayor a 0.
fecha_vencimiento	Fecha límite para realizar el pago	DATE	No nulo
estado	Estado del pago (1 = pagado, 0 = pendiente)	INTEGER	No nulo

Detalles

Relaciones: El campo alumno es la clave foránea que hace referencia a la tabla alumno.

Restricciones: El campo monto debe ser mayor a 0 y de tipo NUMERIC (38,2) para representar valores monetarios. El campo estado solo puede registrar valores 1 (pagado), 0 (pendiente).

5 Inventario de librerías y componentes

5.1 Tecnologías y librerías utilizadas

a. Java 17

Para realizar el reto, he optado por Java debido a su robustez, escalabilidad y amplio soporte. Además, es el lenguaje que más domino, ya que, por las razones mencionadas, decidí profundizar en su aprendizaje y fortalecer mis habilidades en esta tecnología.

b. Spring Boot

Para desarrollar el reto, he optado por usar spring boot por ser un framework con capacidad de simplificar el desarrollo de aplicaciones Java.

c. Maven

He elegido Maven para este reto debido a su gestión eficiente de proyectos y dependencias en Java. Las dependencias usadas para este proyecto son las siguientes:

- Spring boot jpa.
- Spring boot starter web.
- Driver PostgresSQL.
- Lombok
- Jackson fasterXML.
- Spring doc open api.

d. Postgresql

Base de datos muy popular debido a su robustez y capacidad para manejar grandes volúmenes de datos de manera eficiente

e. Neon Tech

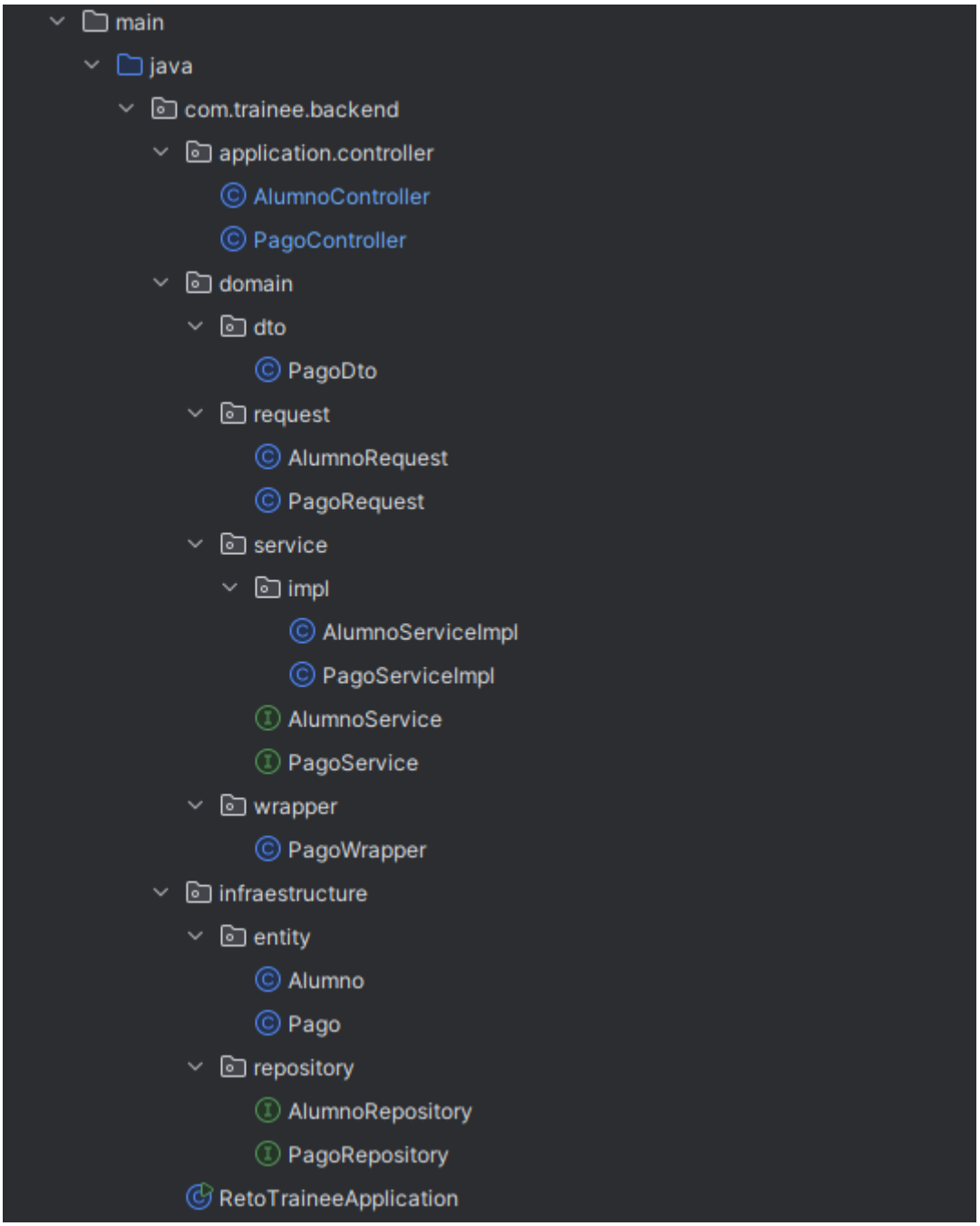
Para desplegar mi base de datos en la nube, usé Neon (*https://neon.tech/home*). Neon es una plataforma de base de datos Postgresql basada en la nube que se enfoca en la escalabilidad y rendimiento.

f. Azure

Para desplegar mi aplicación en la nube, use la alternativa de aplicaciones web en azure debido a su facilidad de uso y su capacidad de escalabilidad

5.2 Componentes

Listado de carpetas de los componentes para realizar el proyecto



Controladores

Se usará 2 controladores: Alumno Controller, para manejar las peticiones relacionadas al alumno, y Pago controller, para manejar las peticiones relacionadas a los pagos.

Dto

Se uso una clase PagoDto, para realizar las respuestas de los pagos.

Request

Se manejo 2 archivos request. PagoRequest y AlumnoRequest, en los cuales se modela el formato de entrada en el body de la petición.

Servicios

Se implemento las interfaces AlumnoService y Pago Service para establecer la lógica de negocio. Aquí se incluyó las validaciones.

Wrapper

Se creo una clase PagoWrapper para personalizar el archivo XML.

Entity

Se creo 2 entidades: Alumno y pago para modelar las tablas en la base de datos.

Repositorios

Se creo 2 repositorios para las entidades con el objetivo de interactuar con la base de datos.

6 Ejecución del proyecto

6.1 Ejecución local

Para ejecutar la aplicación localmente, primero se debe clonar el proyecto desde el repositorio de github. Además, es necesario crear la base de datos `alumno` en PostgreSQL. Posteriormente, se debe configurar las credenciales en el archivo `application.properties` las credenciales de la base de datos

```
spring.datasource.url= ${DATABASE_URL}

spring.datasource.username= ${DATABASE_USERNAME}

spring.datasource.password= ${DATABASE_PASSWORD}
```

Las tablas serán creadas automáticamente desde la aplicación en Spring Boot. Para facilitar el uso de mi aplicación he desplegado la aplicación en la nube de Azure, tal como se detallarán a continuación.

6.2 Ejecución en la Nube

Para ejecutar la aplicación en la nube, solo es necesario instalar Postman o una aplicación similar para probar APIs. La aplicación ha sido desplegada en Azure, y el dominio para acceder a ella es el siguiente:

https://xpedition-challenge-g4bzafb5fja9hbe4.ukwest-01.azurewebsites.net

Para facilitar el uso de mi API, he creado una colección en Postman llamada "xpedition-challenge-cloud", que está disponible para su uso.

7 Catálogo de servicios web implementados

7.1 Registro de alumno

Descripción: Servicio que permite registrar un alumno en la base de datos.

Método: Post

Endpoint: */api/v1/alumnos/crear*

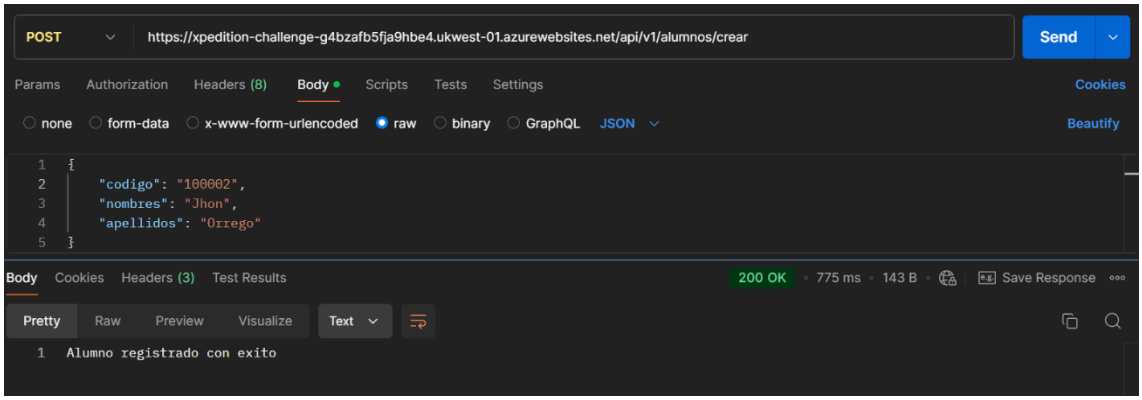
Parámetros: Para registrar un alumno, se debe enviar en el body de la solicitud un JSON de la siguiente forma.

```
{
  "codigo": "string",
  "nombres": "string",
  "apellidos": "string"
}
```

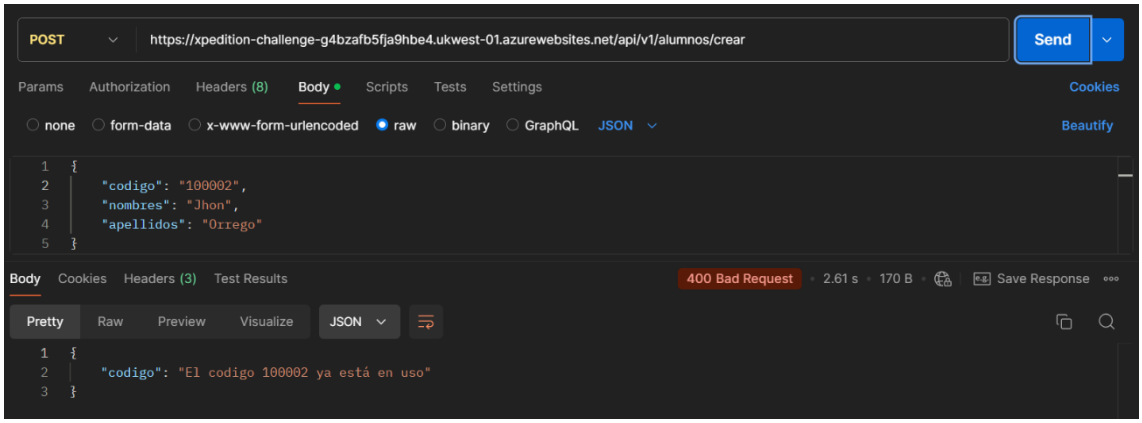
Si la petición es exitosa, se retornará un mensaje indicando el registro exitoso del alumno. Caso contrario, se rechazará la petición indicando el motivo.

Ejemplos:

- Caso exitoso



- Error



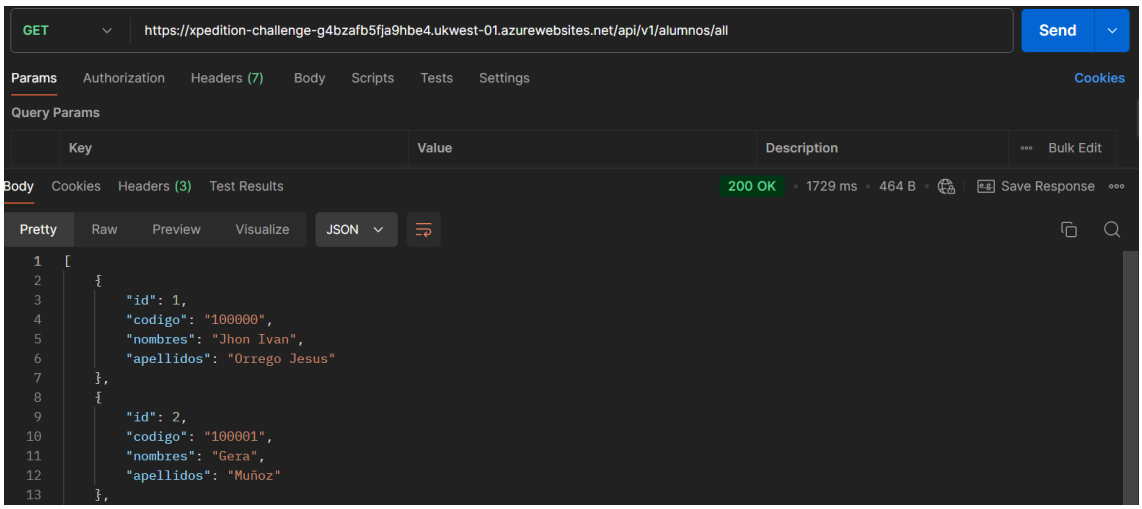
7.2 Listar alumnos

Descripción: Servicio que permite obtener una lista de todos los alumnos de la base de datos.

Método: Get

Endpoint: */api/v1/alumnos/all*

Ejemplo:



7.3 Registrar Pago

Descripción: Servicio que permite registrar un pago en la base de datos

Método: Post

Endpoint*: /api/v1/pagos/registrar-pago*

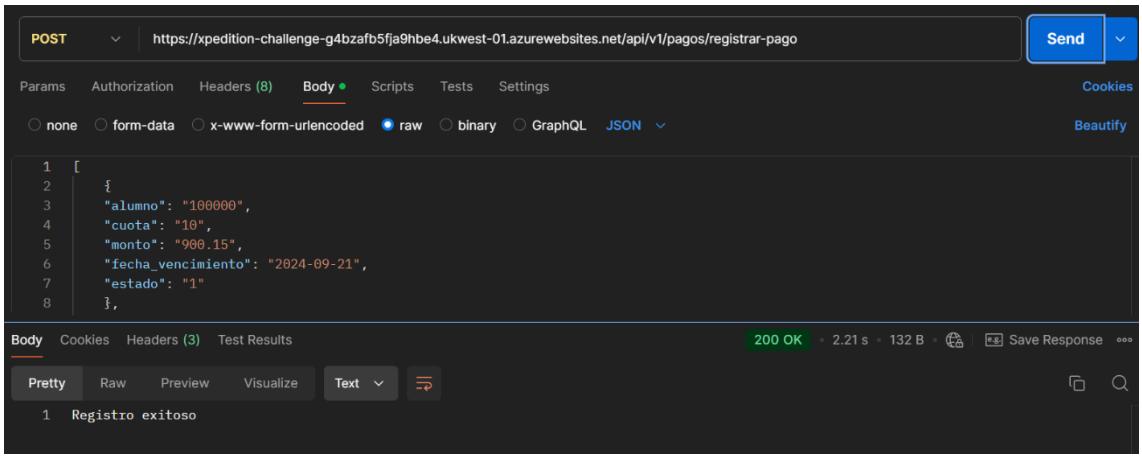
Parámetros: Para registrar un pago, se debe enviar un JSON en el body de la solicitud, un JSON de la siguiente forma:

```
[
  {
    "alumno": "string",
    "cuota": "string",
    "monto": "string",
    "fecha_vencimiento": "string",
    "estado": "string"
  }
]
```

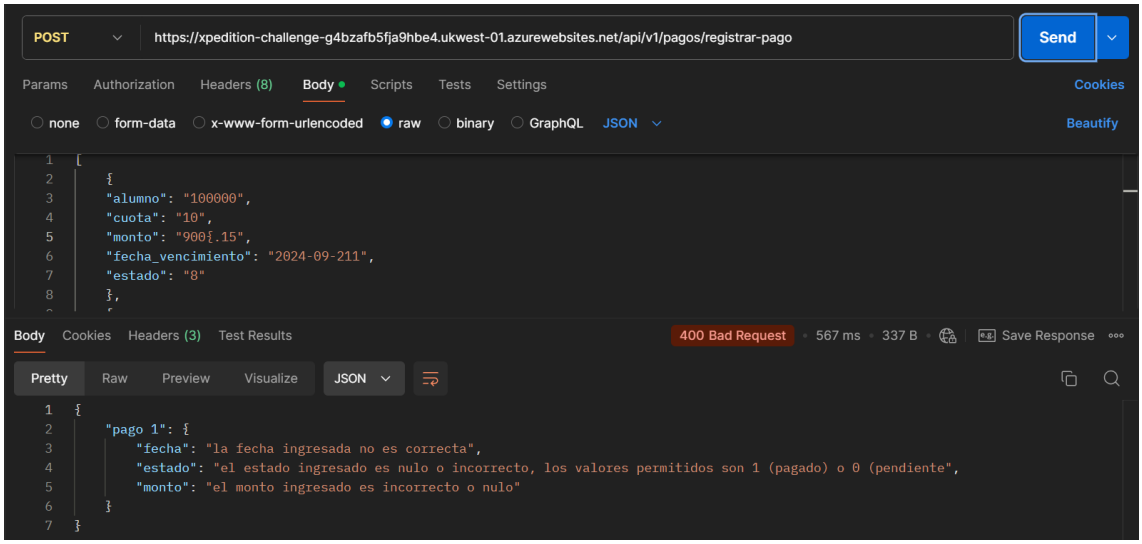
Si la petición es exitosa, se retornará un mensaje indicando el registro exitoso del pago. Caso contrario, se rechazará la petición indicando el motivo.

Ejemplos:

- Caso exitoso



- Error



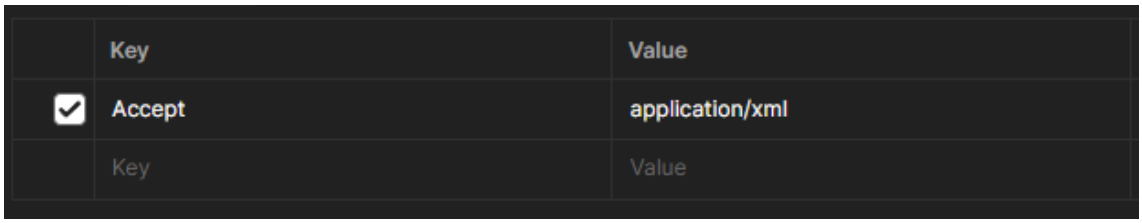
7.4 Listar Pagos

Descripción: Servicio que permite obtener todos los pagos desde la base de datos.

Método: Get

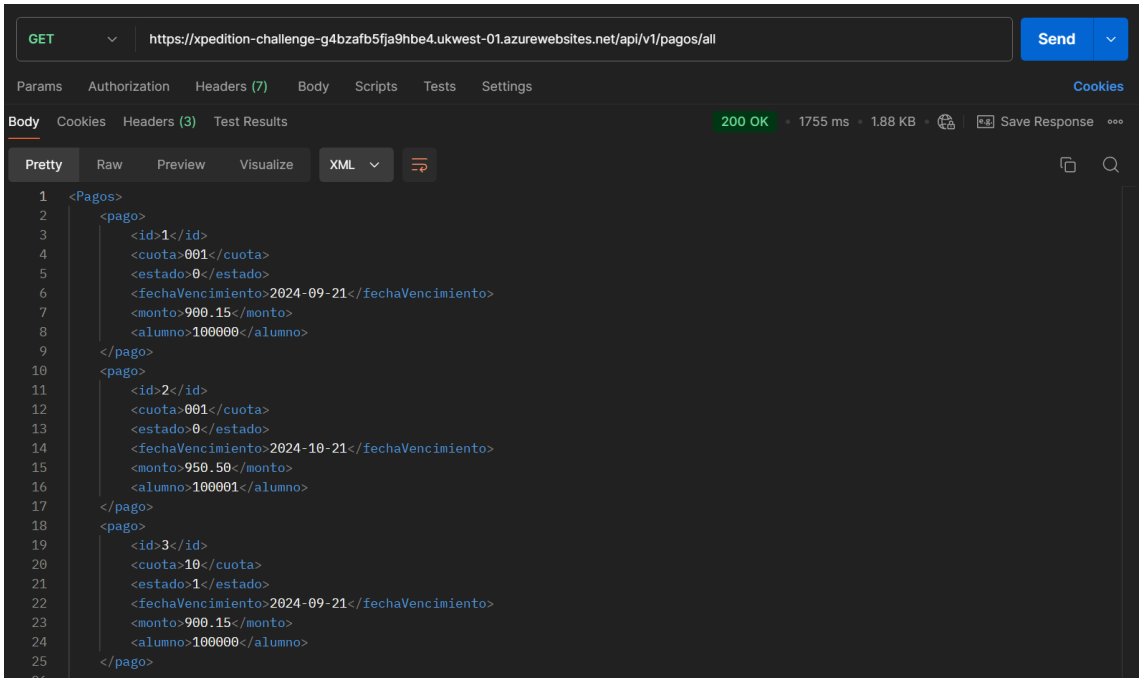
Endpoint: */api/v1/pagos/all*

Se debe añadir la propiedad Accept con valor application/XML en el header para obtener en formato XML solicitado.



Ejemplo:

- Caso exitoso



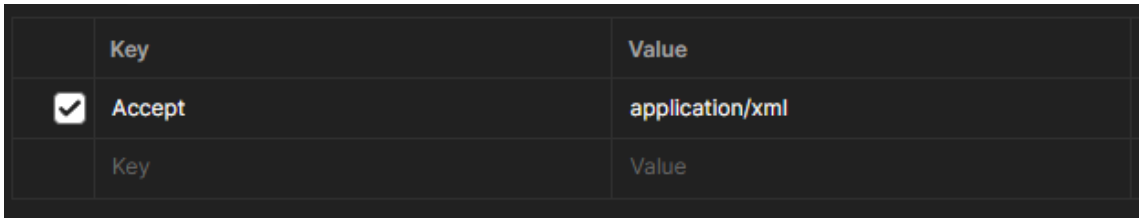
7.5 Listar Pagos Pendientes

Descripción: Servicio que permite obtener todos los pagos pendientes desde la base de datos.

Método: Get

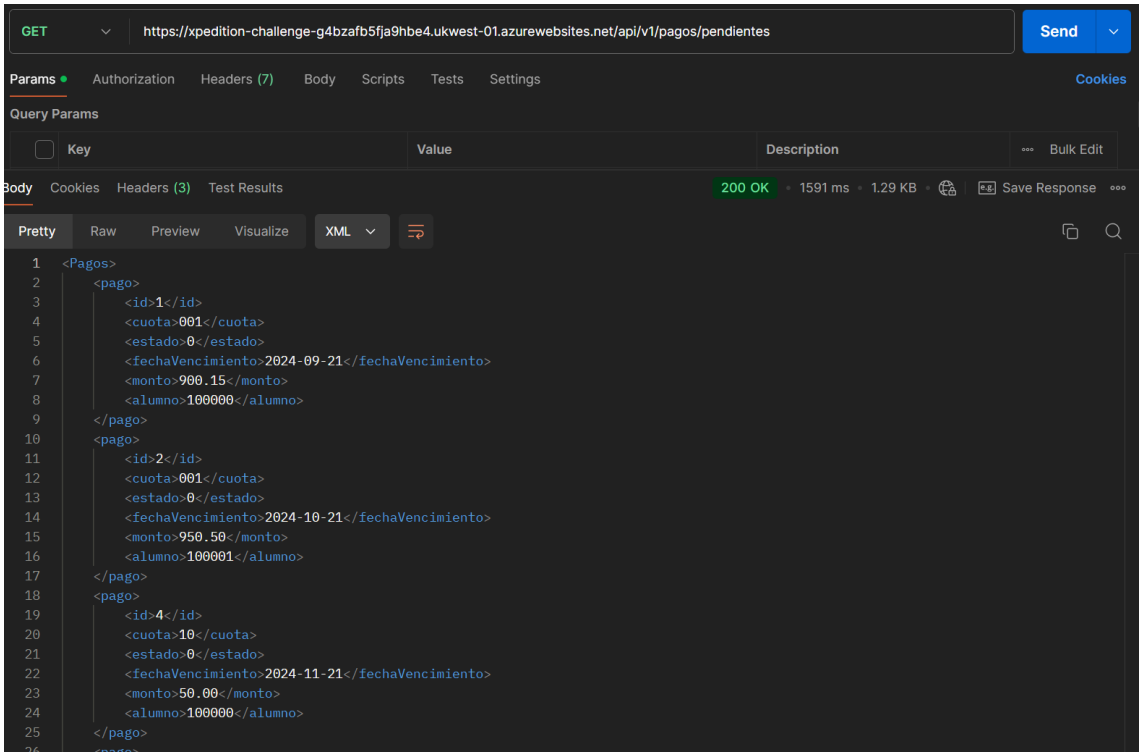
Endpoint: */api/v1/pagos/pendientes*

Se debe añadir la propiedad Accept con valor application/XML en el header para obtener en formato XML solicitado.



Ejemplo:

- Caso exitoso


GET
https://xpedition-challenge-g4bzafb5fja9hbe4.ukwest-01.azurewebsites.net/api/v1/pagos/pendientes
Send
Params
Authorization
Headers (7)
Body
Scripts
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (3)
Test Results
200 OK
1591 ms
1.29 KB
Save Response
Pretty
Raw
Preview
Visualize
XML
<Pagos>
<pago>
<id>1</id>
<cuota>001</cuota>
<estado>0</estado>
<fechaVencimiento>2024-09-21</fechaVencimiento>
<monto>900.15</monto>
<alumno>100000</alumno>
</pago>
<pago>
<id>2</id>
<cuota>001</cuota>
<estado>0</estado>
<fechaVencimiento>2024-10-21</fechaVencimiento>
<monto>950.50</monto>
<alumno>100001</alumno>
</pago>
<pago>
<id>4</id>
<cuota>10</cuota>
<estado>0</estado>
<fechaVencimiento>2024-11-21</fechaVencimiento>
<monto>50.00</monto>
<alumno>100000</alumno>
</pago>