🏛️ UNIVERSIDAD INEXSU

INFORME TÉCNICO — PROYECTO ACADEMIUM

API POTTER's

ENTREGA 2

Presentado por:
Juliana Pérez — Jefferson Perdomo — Santiago Barrera
Fecha: Noviembre 2025

---

1. DESCRIPCIÓN DEL PROYECTO

Academium es una API desarrollada en **Node.js** con **TypeScript** y el framework **Fastify**, diseñada bajo los principios de la **Arquitectura Hexagonal**.
Su objetivo es permitir la gestión académica de entidades principales como **Asignaturas**, **Programas Académicos** y **Períodos Académicos**, garantizando la separación de responsabilidades, la escalabilidad y el mantenimiento del código.

---

2. IMPLEMENTACIONES

Los **CRUD completos** para las entidades:

- Asignatura
- Programa Académico
- Período Académico
- Plan de Estudio
- Oferta

Bajo una **arquitectura limpia y escalable**.
El sistema permite **crear, consultar, actualizar y eliminar** los registros de cada módulo,

validando la información y controlando los posibles errores mediante **códigos de respuesta estandarizados**.

---

3. ARQUITECTURA DEL PROYECTO

El proyecto sigue el modelo de **Arquitectura Hexagonal**, que busca desacoplar las capas de la aplicación para facilitar la mantenibilidad y la extensibilidad del sistema.

3.1 Estructura General del Proyecto

```
Index
 └── app
      └── Rutas
           └── Controladores
                └── Esquemas
                     └── Aplicación (Casos de uso)
                          └── Infraestructura (Repositorio)
                               └── Dominio (Entidades)
```

3.2 Descripción de Capas

- **Index.ts:** Punto de entrada del servidor, encargado de inicializar la aplicación.
- **App.ts:** Configuración general, conexión a base de datos, registro de rutas y middlewares.
- **Rutas:** Define los endpoints de cada módulo.
- **Controladores:** Procesan las solicitudes HTTP y comunican con los casos de uso.
- **Esquemas:** Validan la estructura y tipos de datos recibidos.
- **Aplicación:** Contiene la lógica de negocio (casos de uso).

- **Infraestructura:** Maneja la conexión con la base de datos (repositorios PostgreSQL).
- **Dominio:** Define las entidades y modelos del negocio.

---

4. EJEMPLO DE CONFIGURACIÓN DEL ARCHIVO `.env`

```
# http client

PUERTO=3000


# Base de datos

PGHOST=localhost

PGPORT=5432

PGUSER=”Tu-user”

PGPASSWORD=”Tu-password”

PGDBNAME=bd_academium
```

---

5. DOCUMENTACIÓN DE ENDPOINTS

Prefijo general de los endpoints:
`/api/Academium/`

---

6. PLAN DE ESTUDIO (PROGRAMA ACADÉMICO - ASIGNATURA)

Este servicio permite vincular una asignatura a un programa académico específico, detallando el semestre/nivel en el que debe cursarse y la cantidad de créditos o carga horaria que otorga dentro de ese plan de estudio.
El proceso incluye validaciones rigurosas para asegurar la coherencia de los datos, como

verificar la existencia del programa y la asignatura, evitar duplicidades, y garantizar que los valores de semestre y créditos sean coherentes.

🔹 **Crear un Plan de Estudio (POST)**

Endpoint:

```
POST /api/Academium/planestudio
```

Body (JSON):

```
{
  "idprograma": 3,
  "idasignatura": 5,
  "semestre": 11,
  "creditos": 25
}
```

✅ **Respuesta exitosa:**

```
{
  "mensaje": "El plan de estudio se creó correctamente",
  "idNuevoPlanEstudio": {
    "idplanestudio": 19,
    "nombreprograma": "Especialización en Finanzas",
    "nombreasignatura": "Redes de Computadores",
    "semestre": 11,
    "creditos": 25
  }
}
```

❌ **Casos de error:**

1. **Error de Inexistencia (Programa o Asignatura):**

```
{
  "mensaje": "Error al registrar el plan de estudio",
  "error": "El Programa Académico con ID 999 no existe"
  "error": "La Asignatura con ID 88 no existe"
}
```

2. **Error de Duplicidad:**

```
{
  "mensaje": "Error al registrar el plan de estudio",
  "error": "El plan de estudio ya existe para este programa, asignatura y semestre"
}
```

3. **Error de Valores Inválidos:**

```
{
  "mensaje": "Error al registrar el plan de estudio",
  "error": "No puede tener más de 30 créditos"
  "error": "El semestre máximo permitido es 12"
}
```

🔹 **Consultar todo el Plan de Estudio (GET)**

Endpoint:

`GET /api/Academium/planestudio`

✅ **Respuesta exitosa:**


```
{
  "mensaje": "Plan de estudio encontrado correctamente",
  "PlanEstudio": [
    {
      "idplanestudio": 4,
      "idprograma": 2,
      "idasignatura": 2,
      "semestre": 1,
      "creditos": 3
    },
    {
      "idplanestudio": 6,
      "idprograma": 1,
      "idasignatura": 2,
      "semestre": 11,
      "creditos": 4
    }
  ],
  "planEstudioEncontrados": 2
```

```
}
```

❌ **Error general:**
Este método no presenta errores propiamente dichos; siempre que el endpoint esté correctamente configurado, la solicitud se ejecutará exitosamente.

🔹 **Consultar Plan de Estudio por ID (GET)**

Endpoint:

```
GET /api/Academium/planestudio/:idPlanEstudio
```

✅ **Respuesta exitosa:**

```
{
  "mensaje": "Plan de estudio encontrado correctamente",
  "planEstudio": {
    "idplanestudio": 2,
    "idprograma": 2,
    "idasignatura": 3,
    "semestre": 4,
    "creditos": 15
  }
}
```

⚠️ **Si el ID no existe:**

```
{ "mensaje": "Plan de estudio no encontrada" }
```

🔹 **Actualizar Plan de Estudio (PUT)**

Endpoint:

```
PUT /api/Academium/planestudio/:idPlanEstudio
```

Permite modificar los atributos de un vínculo existente, como el semestre/nivel o los créditos/carga horaria. Este proceso también evita duplicidades, y garantizar que los valores de semestre y créditos sean coherentes.

Body (JSON):

```
{
  "idPrograma": 3,
  "idAsignatura": 4,
  "semestre": 3,
  "creditos": 18
}
```

✅ **Respuesta exitosa:**

```
{
  "mensaje": "Plan de estudio actualizado correctamente",
  "planEstudioActualizado": {
    "idplanestudio": 1,
    "nombreprograma": "Especialización en Finanzas",
    "nombreasignatura": "Bases de Datos",
    "semestre": 3,
    "creditos": 18
  }
}
```

❌ **Casos de error:**

1. **Error de Inexistencia (ID o referencias inexistentes):**

```
{
  "mensaje": "Plan de estudio no encontrado para actualizar",
  "error": "El vínculo con ID [ID_NO_EXISTENTE] no existe."
}
{
  "mensaje": "Error al actualizar el plan de estudio",
  "error": "El Programa Académico con ID 999 no existe"
  "error": "La Asignatura con ID 88 no existe"
}
```

2. **Error de Duplicidad:**

```
{
  "mensaje": "Error al actualizar el plan de estudio",
  "error": "El plan de estudio ya existe con la misma asignatura
y semestre en este programa."
}
```

3. **Error de Valores Inválidos:**

```
{
  "mensaje": "Error al actualizar el plan de estudio",
  "error": "No puede tener más de 30 créditos"
}
```

```
{
  "mensaje": "Error al actualizar el plan de estudio",
  "error": "El semestre máximo permitido es 12"
}
```

🔹 **Eliminar Plan de Estudio (DELETE)**

Endpoint:
`DELETE /api/Academium/planestudio/:idPlanEstudio`

✅ **Respuesta exitosa:**

```
{
  "mensaje": "Plan de estudio eliminado correctamente",
  "idPlanEstudio": "3"
}
```

6.1 GESTIÓN DE PERÍODOS ACADÉMICOS

🔹 1️⃣Crear un Período Académico (POST)

Al momento de crear un nuevo período académico, es fundamental verificar que no exista otro periodo activo cuyas fechas se solapan con las del nuevo registro.
Se debe comprobar que no haya ningún periodo con el campo `idEstado = 2` (Activo) dentro del rango de fechas especificado.

Antes de crear un nuevo periodo, se recomienda utilizar el método GET para consultar los períodos existentes y determinar cuál se encuentra actualmente abierto. De esta manera, se evita la superposición de fechas y estados.

Todo nuevo periodo académico debe iniciarse con el estado **“En preparación” (idEstado = 1)**, garantizando que solo uno se encuentre activo a la vez y que el flujo de transición de estados sea coherente.

Endpoint:

`POST /api/Academium/periodoacademico`

Body (JSON):

```
{
  "semestre": "2025-2",
  "fechaInicio": "2025-07-01",
  "fechaFin": "2025-11-15",
  "idEstado": 1
}
```

✅ **Respuesta exitosa:**

```
{
  "mensaje": "El periodo académico se creó correctamente",
  "idNuevoPeriodo": {
    "idperiodo": 36,
    "semestre": "2025-2",
    "fechainicio": "2025-07-01",
    "fechafin": "2025-11-15",
    "estadoperiodo": "preparacion"
  }
}
```

❌ **Casos de error:**


```
{
  "mensaje": "No se puede crear el período académico",
  "error": "Existe un periodo activo que solapa las fechas. idPeriodo 1 periodo desde 2025-01-14 hasta 2025-06-14"
}
{
  "mensaje": "No se puede crear el período académico",
  "error": "La fecha de inicio no puede ser mayor que la fecha de fin"
}
```

🔹 2️⃣ Consultar Todos los Períodos Académicos (GET)

Al realizar la consulta mediante el método GET, se obtiene un listado de todos los periodos académicos registrados en la base de datos. Cada registro incluye su identificador, semestre, fecha de inicio, fecha de finalización y el estado actual en el que se encuentra.

Esta funcionalidad permite tener una visión general y actualizada de los periodos académicos, facilitando la gestión, el control y la toma de decisiones dentro del sistema.

Endpoint:
`GET /api/Academium/periodoacademico`

✅ **Respuesta exitosa:**


```
{
  "mensaje": "Periodos académicos encontrados correctamente",
  "periodos": [
    {
```

```
      "idperiodo": 1,
      "semestre": "2025-1",
      "fechainicio": "2025-01-15",
      "fechafin": "2025-06-15",
      "idestado": 2
    },
    {
      "idperiodo": 2,
      "semestre": "2025-2",
      "fechainicio": "2025-07-01",
      "fechafin": "2025-12-15",
      "idestado": 1
    }
  ],
  "periodosEncontrados": 2
}
```


❌ **Error general:**
No presenta errores propiamente dichos, ya que su función es solo de consulta.

🔹 3️⃣Consultar Período Académico por ID (GET)

Endpoint:
`GET /api/Academium/periodoacademico/:idPeriodo`

✅ **Respuesta exitosa:**


```
{
```

```
  "mensaje": "Periodo académico encontrado correctamente",
  "periodo": {
    "idperiodo": 1,
    "semestre": "2025-1",
    "fechainicio": "2025-01-15",
    "fechafin": "2025-06-15",
    "idEstado": 2
  }
}
```


⚠️ **Si el ID no existe:**


```
{ "mensaje": "Periodo académico no encontrada" }
```

🔹 4️⃣ Actualizar Período Académico (PUT)

Endpoint:
`PUT /api/Academium/periodoacademico/:idPeriodo`

Body (JSON):


```
{
  "semestre": "2025-2",
  "fechaInicio": "2025-07-01",
  "fechaFin": "2025-11-15",
  "idEstado": 2
}
```

✅ **Respuesta exitosa:**


```
{
  "mensaje": "Periodo académico actualizado correctamente",
  "periodoActualizado": {
    "idperiodo": 36,
    "semestre": "2025-2",
    "fechainicio": "2025-07-01",
    "fechafin": "2025-11-15",
    "estadoperiodo": "activo"
  }
}
```


❌ **Casos de error:**

Si no se respeta la secuencia lógica de transición entre estados —Preparación → Activo → Cerrado— al intentar actualizar el estado de un periodo académico, el sistema generará un error. Esta validación garantiza la coherencia del flujo y evita inconsistencias en la gestión de los periodos académicos.


```
{
  "mensaje": "Error al actualizar el periodo académico",
  "error": "Transición de estado no permitida: no se puede cambiar de Activo a Preparación"
}
```


Los campos del body deben seguir una secuencia y formato preestablecidos camelCase.

Semestre: debe incluir año y número de semestre (formato AAAA-N).

Fechas: deben cumplir el formato AAAA-MM-DD y ser coherentes; la fecha de fin no puede ser anterior a la de inicio.

Estado: solo se permiten valores válidos (1, 2 o 3).

Si alguno de estos campos no cumple con el formato o la secuencia lógica, el sistema generará errores de validación.

```
{

  "mensaje": "Error de validación al actualizar un nuevo periodo académico",

  "error": "El formato de la fecha de inicio es inválido. Debe ser AAAA-MM-DD"

"error": "Debe tener al menos 6 caracteres con un formato año-semestre (ej: 2025-2)"

 "error": "La fecha de inicio no puede ser mayor que la fecha de fin"

  "error": "El idEstado sólo puede ser 1, 2 o 3"

}
```

🔹 5️⃣ Eliminar Período Académico (DELETE)

Endpoint:
`DELETE /api/periodoacademico/:idPeriodo`

✅ **Respuesta exitosa:**

```
{

  "mensaje": "Periodo académico eliminado correctamente",

  "idPeriodo": "37"

}
```

6.2 GESTIÓN DE OFERTAS

🔹 1️⃣Crear una Oferta (POST)

Al crear una nueva oferta académica, es fundamental verificar que los datos correspondan a registros válidos y coherentes dentro del sistema. Para ello, se debe comprobar la existencia del periodo académico, del programa y de la asignatura asociados antes de proceder con el registro.

El periodo académico debe encontrarse activo (idEstado = 2), ya que únicamente en este estado se permite generar nuevas ofertas. Además, se debe validar que no exista duplicidad de grupos, evitando crear dos ofertas con el mismo identificador para una misma asignatura y programa dentro del mismo periodo.

Por último, el campo cupo debe ser un número entero mayor que cero, garantizando la disponibilidad de plazas para los estudiantes. Cumplidas estas condiciones, el sistema registrará la nueva oferta académica y confirmará su creación con los datos esenciales del grupo creado.

Endpoint:

```
POST /api/Academium/ofertas
```

Body (JSON):

```
{

  "idPrograma": 2,

  "idPeriodo": 1,

  "idAsignatura": 3,

  "grupo": 1,

  "cupo": 30

}
```

✅ **Respuesta exitosa:**

```
{
  "mensaje": "La oferta se creó correctamente",
  "ofertaCreada": {
    "idoferta": 8,
    "nombreprograma": "Administración de Empresas",
    "semestre": "2025-1",
    "nombreasignatura": "Laboratorio de Física",
    "informacion": "Prácticas experimentales de mecánica y electricidad."
  }
}
```

❌ **Casos de error:**

```
{
  "mensaje": "Error al crear una nueva oferta",
  "error": "Ya existe un grupo matriculado con la misma asignatura, programa y periodo académico"
  "error": "El periodo está en preparación"
}
```

🔹 2️⃣Consultar Todas las Ofertas (GET)

Al realizar la consulta mediante el método GET, se obtiene un listado de todas las ofertas académicas registradas en la base de datos. Cada registro incluye su identificador, identificador del programa, identificador del período, identificador de la asignatura, el grupo y la cantidad de cupo que se dispone.

Esta funcionalidad permite tener una visión general y actualizada de los periodos académicos, facilitando la gestión, el control y la toma de decisiones dentro del sistema.

Endpoint:
`GET /api/Academium/ofertas`

✅ **Respuesta exitosa:**


```
{
  "mensaje": "Ofertas encontradas correctamente",
  "Ofertas": [
    {
      "idoferta": 1,
      "idprograma": 1,
      "idperiodo": 1,
      "idasignatura": 1,
      "grupo": 1,
      "cupo": 30
    },
    {
```

```
      "idoferta": 2,
      "idprograma": 1,
      "idperiodo": 1,
      "idasignatura": 2,
      "grupo": 1,
      "cupo": 25
    }
  ],
  "ofertasEncontrados": 2
}
```


❌ **Error general:**
Este método no presenta errores propiamente dichos, ya que su función principal es consultar los registros existentes en la base de datos. Siempre que el endpoint esté correctamente configurado, la solicitud se ejecutará de manera exitosa y retornará todas las ofertas almacenadas.

🔹 3️⃣Consultar Oferta por ID (GET)

Endpoint:
`GET /api/Academium/ofertas/:idOferta`

✅ **Respuesta exitosa:**


```
{
  "mensaje": "Oferta encontrada correctamente",
  "Oferta": {
    "idoferta": 1,
```

```
    "idprograma": 1,
    "idperiodo": 1,
    "idasignatura": 1,
    "grupo": 1,
    "cupo": 30
  }
}
```

⚠️ **Si el ID no existe:**

```
{ "mensaje": "Oferta no encontrada" }
```

🔹 4️⃣ Actualizar Oferta (PUT)

Endpoint:
`PUT /api/Academium/ofertas/:idOferta`

Body (JSON):

```
{
  "idPrograma": 1,
  "idPeriodo": 1,
  "idAsignatura": 4,
  "grupo": 1,
  "cupo": 30
}
```

✅ **Respuesta exitosa:**

```
{
  "mensaje": "Oferta actualizada correctamente",
  "OfertaActualizada": {
    "idoferta": 6,
    "nombreprograma": "Ingeniería de Sistemas",
    "semestre": "2025-1",
    "nombreasignatura": "Bases de Datos",
    "información": "Teoría y práctica del diseño y manejo de bases de datos relacionales."
  }
}
```

❌ **Casos de error:**

Al actualizar una oferta académica, es necesario verificar la existencia previa de la asignatura, el periodo académico y el programa académico relacionados. Si el sistema detecta que ya existe una oferta con el mismo grupo asignado a la misma asignatura, periodo y programa, se generará un error de duplicidad, impidiendo la actualización.

Esta validación garantiza la integridad de la información y evita conflictos en el registro de las ofertas académicas.

```
{
  "mensaje": "Error al actualizar la oferta",
  "error": "Ya existe un grupo matriculado con la misma asignatura, programa y periodo académico"
  "error": "El periodo está en preparación"
  "error": "No se encontró la asignatura ingresada"
```

```
  "error": "No se encontró el periodo ingresado"

  "error": "El cupo debe tener al menos 10 estudiantes"

}
```

🔹 5️⃣ Eliminar Oferta (DELETE)

Endpoint:

```
DELETE /api/ofertas/:idOferta
```

✅ **Respuesta exitosa:**

```
{

  "mensaje": "Oferta eliminada correctamente",

  "idOferta": "6"

}
```

📘 CONCLUSIÓN

El desarrollo del proyecto Academium bajo el modelo de Arquitectura Hexagonal permite una gestión modular y mantenible de los componentes académicos.
Los endpoints documentados proporcionan una comunicación clara y estandarizada entre el cliente y el servidor, garantizando control de errores, validaciones estrictas y escalabilidad del sistema.