Acceso a datos

ActUT05. Creación de servicios web con Spring Boot

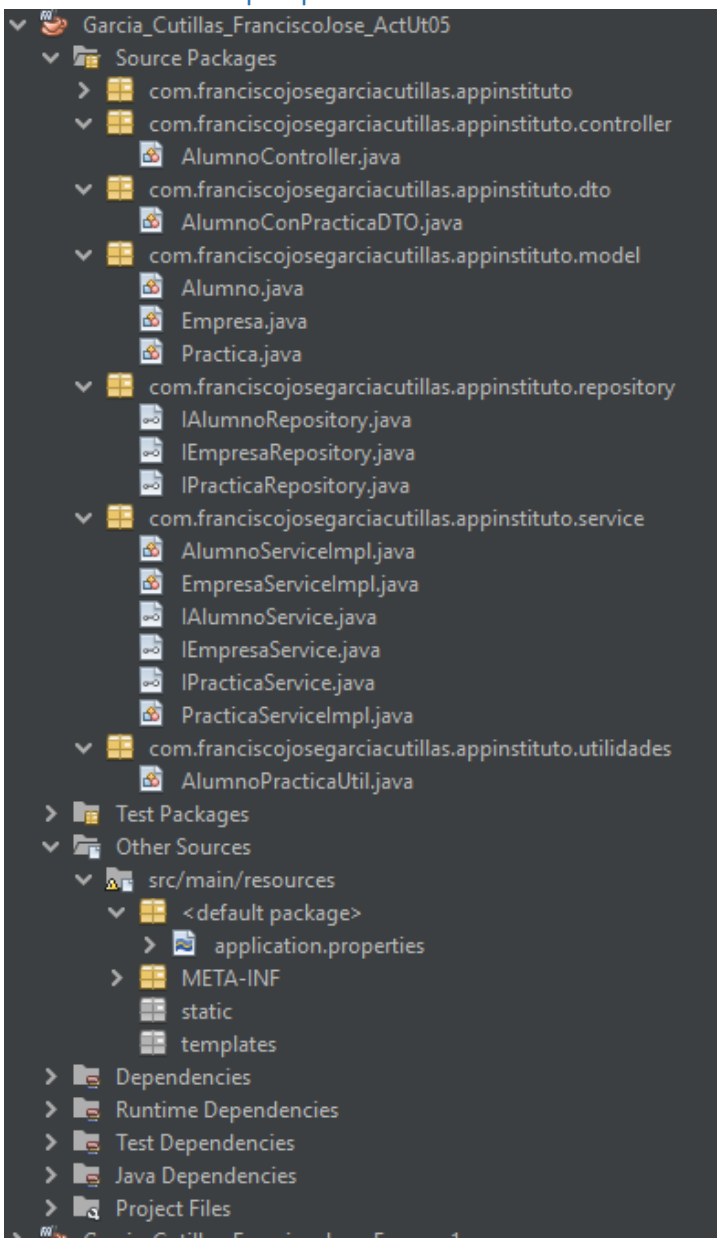
Francisco José García Cutillas | 2FPGS_DAM

Índice

Properties

```
spring.jpa.database=mysql
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql: true

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/fct
spring.datasource.username=root
spring.datasource.password=root

```

Estructura de paquetes



Appinstituto

```java
package com.franciscojosegarciacutillas.appinstituto;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class AppinstitutoApplication {

    public static void main(String[] args) {
        SpringApplication.run( primarySource: AppinstitutoApplication.class, args);
    }

}

```

Controller

```java
package com.franciscojosegarciacutillas.appinstituto.controller;

import com.franciscojosegarciacutillas.appinstituto.dto.AlumnoConPracticaDTO;
import com.franciscojosegarciacutillas.appinstituto.model.Alumno;
import com.franciscojosegarciacutillas.appinstituto.model.Practica;
import com.franciscojosegarciacutillas.appinstituto.service.AlumnoServiceImpl;
import com.franciscojosegarciacutillas.appinstituto.service.PracticaServiceImpl;
import com.franciscojosegarciacutillas.appinstituto.utilidades.AlumnoPracticaUtil;
import java.util.ArrayList;
import java.util.Objects;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 *
 * @author Fran
 */
@RestController
@RequestMapping("/api/v1")
public class AlumnoController {

    @Autowired
    AlumnoServiceImpl aluServ;

    @Autowired
    PracticaServiceImpl pracServ;

    //1.Recuperar todos los alumnos
    @GetMapping("/alumnos")
    public ResponseEntity<ArrayList<AlumnoConPracticaDTO>> getAlumnosConPractica() {

        ArrayList<Alumno> alumnos = aluServ.listar();
        ArrayList<AlumnoConPracticaDTO> alumnosDevolver = new ArrayList<>();

        for (Alumno alumno : alumnos) {

            AlumnoConPracticaDTO nuevoAlumno = new AlumnoConPracticaDTO(alumno);
            alumnosDevolver.add( e: nuevoAlumno);

        }

        return new ResponseEntity<>( body: alumnosDevolver,  status: HttpStatus.OK);

    }

    //2.Recuperar los datos de un alumno en concreto
    @GetMapping("/alumnos/{id}")
    public ResponseEntity<Alumno> getAlumnoId(@PathVariable Integer id) {

        Alumno alumno = aluServ.obtener(id);

```

```
        if (alumno.getAlumnoId() == null) {

            return new ResponseEntity<>(status:HttpStatus.NOT_FOUND);

        } else {

            return new ResponseEntity<>(body:alumno, status:HttpStatus.OK);

        }

    }

    //3.Insertar un alumno sin práctica
    @PostMapping("/alumnos")
    public ResponseEntity<Alumno> postAlumnoSinPractica(@RequestBody Alumno alumno) {

        if (alumno.getAlumnoId()!=null){
            Alumno alumnoExiste = aluServ.obtener(id:alumno.getAlumnoId());
            if (alumnoExiste.getAlumnoId()!=null){
                return new ResponseEntity<>(status:HttpStatus.CONFLICT);
            }
        }

        Alumno alumnoInsertar = aluServ.registrar(alumno);

        return new ResponseEntity<>(body:alumnoInsertar, status:HttpStatus.CREATED);

    }

    //4.Actualizar datos de un alumno
    @PutMapping("/alumnos/{id}")
    public ResponseEntity<Alumno> putAlumno(@RequestBody Alumno alumno, @PathVariable Integer id) {

        Alumno alumnoModificar = aluServ.obtener(id);

        if (alumnoModificar.getAlumnoId() == null) {

            return new ResponseEntity<>(status:HttpStatus.NOT_FOUND);

        } else {
            if (!Objects.equals(a:alumno.getAlumnoId(), b:id) && alumno.getAlumnoId() != null) {

                return new ResponseEntity<>(status:HttpStatus.BAD_REQUEST);

            } else {

                alumnoModificar.setNombreAlumno(nombreAlumno:alumno.getNombreAlumno());
                alumnoModificar.setEdad(edad:alumno.getEdad());
                alumnoModificar.setTitulo(titulo:alumno.getTitulo());

                alumnoModificar = aluServ.modificar(alumno:alumnoModificar);

            }
        }

        return new ResponseEntity<>(body:alumnoModificar, status:HttpStatus.OK);

    }
```

```
    //5.Insertar una práctica para un alumno
    @PutMapping("/alumnos/practica")
    public ResponseEntity<Practica> putPracticaAlumno(@RequestBody AlumnoPracticaUtil alumnoPracticaUtil){

        Practica practicaInsertar = pracServ.obtener(id:alumnoPracticaUtil.getPracticaId());
        Alumno alumnoPractica = aluServ.obtener(id:alumnoPracticaUtil.getAlumnoId());

        if (practicaInsertar.getPracticaId() == null || alumnoPractica.getAlumnoId() == null) {

            return new ResponseEntity<>(status:HttpStatus.NOT_FOUND);

        }

        practicaInsertar.setAlumnoId(alumnoId:alumnoPractica);
        practicaInsertar = pracServ.modificar(practica:practicaInsertar);

        return new ResponseEntity<>(body:practicaInsertar, status:HttpStatus.OK);

    }

    //6.Eliminar un alumno
    @DeleteMapping("/alumnos/{id}")
    public ResponseEntity<Alumno> deleteAlumno(@PathVariable Integer id) {

        Alumno alumno = aluServ.obtener(id);
        if (alumno.getAlumnoId()== null) {

            return new ResponseEntity<>(status:HttpStatus.NOT_FOUND);

        }

        aluServ.eliminar(id);
        return new ResponseEntity<>(status:HttpStatus.NO_CONTENT);

    }

    //7.Devolver todos los alumnos con su práctica de un determinado titulo
    @GetMapping("/alumnos/titulo/{titulo}")
    public ResponseEntity<ArrayList<AlumnoConPracticaDTO>> getAlumnoTitulo(@PathVariable String titulo){

        ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosTitulo(titulo);
        ArrayList<AlumnoConPracticaDTO> alumnosDevolver = new ArrayList<>();

        if (alumnos.isEmpty()) {

            return new ResponseEntity<>(status:HttpStatus.NOT_FOUND);

        }

        for (Alumno alumno : alumnos) {

            AlumnoConPracticaDTO nuevoAlumno = new AlumnoConPracticaDTO(alumno);
            alumnosDevolver.add(e:nuevoAlumno);

        }

        return new ResponseEntity<>(body:alumnosDevolver, status:HttpStatus.OK);
    }
```

```java
        //8.Devolver todos los alumnos que tienen menos de cierta edad
        @GetMapping("/alumnos/edad/{edad}")
        public ResponseEntity<ArrayList<Alumno>> getAlumnoEdadMenorQue(@PathVariable Integer edad){

            ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosEdadMenorQue(edad);

            if (alumnos.isEmpty()) {

                return new ResponseEntity<>( status:HttpStatus.NOT_FOUND);

            }

            return new ResponseEntity<>( body:alumnos,  status:HttpStatus.OK);

        }

        //9.Obtener todos los alumnos que no tienen práctica
        @GetMapping("/alumnos/sinpractica")
        public ResponseEntity<ArrayList<Alumno>> getAlumnosSinPractica(){

            ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosSinPractica();

            if (alumnos.isEmpty()) {

                return new ResponseEntity<>( status:HttpStatus.NOT_FOUND);

            }

            return new ResponseEntity<>( body:alumnos,  status:HttpStatus.OK);

        }

        //10.Obtener todos los alumnos que tienen práctica asignada
        @GetMapping("/alumnos/conpractica")
        public ResponseEntity<ArrayList<Alumno>> getAlumnosConPracticaAsignada(){

            ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosConPractica();

            if (alumnos.isEmpty()) {

                return new ResponseEntity<>( status:HttpStatus.NOT_FOUND);

            }

            return new ResponseEntity<>( body:alumnos,  status:HttpStatus.OK);

        }

    }
```

Model

```java
package com.franciscojosegarciacutillas.appinstituto.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import java.io.Serializable;
import java.util.List;
import jakarta.persistence.*;

/**
 *
 * @author Fran
 */
@Entity
@Table(name = "alumnos")
@NamedQueries({
    @NamedQuery(name = "Alumno.findAll", query = "SELECT a FROM Alumno a"),
    @NamedQuery(name = "Alumno.findByAlumnoId", query = "SELECT a FROM Alumno a WHERE a.alumnoId = :alumnoId"),
    @NamedQuery(name = "Alumno.findByNombreAlumno", query = "SELECT a FROM Alumno a WHERE a.nombreAlumno = :nombreAlumno"),
    @NamedQuery(name = "Alumno.findByEdad", query = "SELECT a FROM Alumno a WHERE a.edad = :edad"),
    @NamedQuery(name = "Alumno.findByTitulo", query = "SELECT a FROM Alumno a WHERE a.titulo = :titulo")})
public class Alumno implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "alumno_id")
    private Integer alumnoId;
    @Basic(optional = false)
    @Column(name = "nombre_alumno")
    private String nombreAlumno;
    @Basic(optional = false)
    @Column(name = "edad")
    private int edad;
    @Basic(optional = false)
    @Column(name = "titulo")
    private String titulo;
    @JoinTable(name = "alumnoscandpracticas", joinColumns = {
        @JoinColumn(name = "alumno_id", referencedColumnName = "alumno_id")}, inverseJoinColumns = {
        @JoinColumn(name = "practica_id", referencedColumnName = "practica_id")})
    @ManyToMany
    @JsonIgnore
    private List<Practica> practicaList;
    @OneToOne(mappedBy = "alumnoId")
    @JsonIgnore
    private Practica practica;

    public Alumno() {
    }

    public Alumno(Integer alumnoId) {
        this.alumnoId = alumnoId;
    }

    public Alumno(Integer alumnoId, String nombreAlumno, int edad, String titulo) {
        this.alumnoId = alumnoId;
        this.nombreAlumno = nombreAlumno;
        this.edad = edad;
        this.titulo = titulo;
    }

```

```java
    public Alumno(String nombreAlumno, int edad, String titulo) {
        this.nombreAlumno = nombreAlumno;
        this.edad = edad;
        this.titulo = titulo;
    }

    public Integer getAlumnoId() {
        return alumnoId;
    }

    public void setAlumnoId(Integer alumnoId) {
        this.alumnoId = alumnoId;
    }

    public String getNombreAlumno() {
        return nombreAlumno;
    }

    public void setNombreAlumno(String nombreAlumno) {
        this.nombreAlumno = nombreAlumno;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public List<Practica> getPracticaList() {
        return practicaList;
    }

    public void setPracticaList(List<Practica> practicaList) {
        this.practicaList = practicaList;
    }

    public Practica getPractica() {
        return practica;
    }

    public void setPractica(Practica practica) {
        this.practica = practica;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (alumnoId != null ? alumnoId.hashCode() : 0);
        return hash;
    }

```

```java
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Alumno)) {
            return false;
        }
        Alumno other = (Alumno) object;
        if ((this.alumnoId == null && other.alumnoId != null) || (this.alumnoId != null && !this.alumnoId.equals( obj: other.alumnoId))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Alumno con ID = " + alumnoId + ", Nombre = " + nombreAlumno + ", Edad = " + edad + ", Titulo = " + titulo;
    }

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import java.io.Serializable;
import java.util.List;
import jakarta.persistence.*;

/**
 *
 * @author Fran
 */
@Entity
@Table(name = "empresas")
@NamedQueries({
    @NamedQuery(name = "Empresa.findAll", query = "SELECT e FROM Empresa e"),
    @NamedQuery(name = "Empresa.findByEmpresaId", query = "SELECT e FROM Empresa e WHERE e.empresaId = :empresaId"),
    @NamedQuery(name = "Empresa.findByNombreEmpresa", query = "SELECT e FROM Empresa e WHERE e.nombreEmpresa = :nombreEmpresa"),
    @NamedQuery(name = "Empresa.findBySector", query = "SELECT e FROM Empresa e WHERE e.sector = :sector")})
public class Empresa implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "empresa_id")
    private Integer empresaId;
    @Basic(optional = false)
    @Column(name = "nombre_empresa")
    private String nombreEmpresa;
    @Basic(optional = false)
    @Column(name = "sector")
    private String sector;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "empresa")
    @JsonIgnore
    private List<Practica> practicaList;

    public Empresa() {
    }

    public Empresa(Integer empresaId) {
        this.empresaId = empresaId;
    }

    public Empresa(Integer empresaId, String nombreEmpresa, String sector) {
        this.empresaId = empresaId;
        this.nombreEmpresa = nombreEmpresa;
        this.sector = sector;
    }

    public Integer getEmpresaId() {
        return empresaId;
    }

    public void setEmpresaId(Integer empresaId) {
        this.empresaId = empresaId;
    }

    public String getNombreEmpresa() {
        return nombreEmpresa;
    }

    public void setNombreEmpresa(String nombreEmpresa) {
        this.nombreEmpresa = nombreEmpresa;
    }

    public String getSector() {
        return sector;
    }

    public void setSector(String sector) {
        this.sector = sector;
    }

    public List<Practica> getPracticaList() {
        return practicaList;
    }

    public void setPracticaList(List<Practica> practicaList) {
        this.practicaList = practicaList;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (empresaId != null ? empresaId.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Empresa)) {
            return false;
        }
        Empresa other = (Empresa) object;
        if ((this.empresaId == null && other.empresaId != null) || (this.empresaId != null && !this.empresaId.equals(obj:other.empresaId))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Empresa con ID = " + empresaId + ", Nombre = " + nombreEmpresa + ", Sector = " + sector;
    }

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import java.io.Serializable;
import java.util.Date;
import java.util.List;
import jakarta.persistence.*;

/**
 *
 * @author Fran
 */
@Entity
@Table(name = "practicas")
@NamedQueries({
    @NamedQuery(name = "Practica.findAll", query = "SELECT p FROM Practica p"),
    @NamedQuery(name = "Practica.findByPracticaId", query = "SELECT p FROM Practica p WHERE p.practicaId = :practicaId"),
    @NamedQuery(name = "Practica.findByFechaInicio", query = "SELECT p FROM Practica p WHERE p.fechaInicio = :fechaInicio"),
    @NamedQuery(name = "Practica.findByFechaFin", query = "SELECT p FROM Practica p WHERE p.fechaFin = :fechaFin"),
    @NamedQuery(name = "Practica.findByDescripcion", query = "SELECT p FROM Practica p WHERE p.descripcion = :descripcion")})
public class Practica implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "practica_id")
    private Integer practicaId;
    @Basic(optional = false)
    @Column(name = "fecha_inicio")
    @Temporal(TemporalType.DATE)
    private Date fechaInicio;
    @Basic(optional = false)
    @Column(name = "fecha_fin")
    @Temporal(TemporalType.DATE)
    private Date fechaFin;
    @Basic(optional = false)
    @Column(name = "descripcion")
    private String descripcion;
    @ManyToMany(mappedBy = "practicaList")
    @JsonIgnore
    private List<Alumno> alumnoList;
    @JoinColumn(name = "alumno_id", referencedColumnName = "alumno_id")
    @ManyToOne
    @JsonIgnore
    private Alumno alumnoId;
    @JoinColumn(name = "empresa_id", referencedColumnName = "empresa_id")
    @ManyToOne(optional = false)
    private Empresa empresa;

    public Practica() {
    }

    public Practica(Integer practicaId) {
        this.practicaId = practicaId;
    }

    public Practica(Integer practicaId, Date fechaInicio, Date fechaFin, String descripcion) {
        this.practicaId = practicaId;
        this.fechaInicio = fechaInicio;
        this.fechaFin = fechaFin;
        this.descripcion = descripcion;
    }
```

```java
    public Integer getPracticaId() {
        return practicaId;
    }

    public void setPracticaId(Integer practicaId) {
        this.practicaId = practicaId;
    }

    public Date getFechaInicio() {
        return fechaInicio;
    }

    public void setFechaInicio(Date fechaInicio) {
        this.fechaInicio = fechaInicio;
    }

    public Date getFechaFin() {
        return fechaFin;
    }

    public void setFechaFin(Date fechaFin) {
        this.fechaFin = fechaFin;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    public List<Alumno> getAlumnoList() {
        return alumnoList;
    }

    public void setAlumnoList(List<Alumno> alumnoList) {
        this.alumnoList = alumnoList;
    }

    public Alumno getAlumnoId() {
        return alumnoId;
    }

    public void setAlumnoId(Alumno alumnoId) {
        this.alumnoId = alumnoId;
    }

    public Empresa getEmpresa() {
        return empresa;
    }

    public void setEmpresa(Empresa empresaId) {
        this.empresa = empresaId;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (practicaId != null ? practicaId.hashCode() : 0);
        return hash;
    }
```

```java
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Practica)) {
            return false;
        }
        Practica other = (Practica) object;
        if ((this.practicaId == null && other.practicaId != null) || (this.practicaId != null && !this.practicaId.equals(obj:other.practicaId))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Practica con ID = " + practicaId + ", Fecha de inicio = " + fechaInicio + ", Fecha de fin = " + fechaFin +
                ", Descripción = " + descripcion;
    }

}
```

Repository

```java
package com.franciscojosegarciacutillas.appinstituto.repository;

import com.franciscojosegarciacutillas.appinstituto.model.Alumno;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

/**
 *
 * @author Fran
 */
public interface IAlumnoRepository extends JpaRepository<Alumno, Integer> {

    List<Alumno> findByTitulo(String titulo);
    List<Alumno> findByEdadLessThan(Integer edad);
    @Query("FROM Alumno a WHERE a.practica is null")
    List<Alumno> findByPracticaNull();
    @Query("FROM Alumno a WHERE a.practica is not null")
    List<Alumno> findByPracticaNotNull();

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.repository;

import com.franciscojosegarciacutillas.appinstituto.model.Empresa;
import org.springframework.data.jpa.repository.JpaRepository;

/**
 *
 * @author Fran
 */
public interface IEmpresaRepository extends JpaRepository<Empresa, Integer> {

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.repository;

import com.franciscojosegarciacutillas.appinstituto.model.Practica;
import org.springframework.data.jpa.repository.JpaRepository;

/**
 *
 * @author Fran
 */
public interface IPracticaRepository extends JpaRepository<Practica, Integer> {

}
```

Service

```java
package com.franciscojosegarciacutillas.appinstituto.service;

import com.franciscojosegarciacutillas.appinstituto.model.Alumno;
import java.util.ArrayList;

/**
 *
 * @author Fran
 */
public interface IAlumnoService {

    Alumno registrar(Alumno alumno);
    Alumno modificar(Alumno alumno);
    ArrayList<Alumno> listar();
    Alumno obtener(Integer id);
    void eliminar(Integer id);
    ArrayList<Alumno> mostrarAlumnosTitulo(String titulo);
    ArrayList<Alumno> mostrarAlumnosEdadMenorQue(Integer edad);
    ArrayList<Alumno> mostrarAlumnosSinPractica();
    ArrayList<Alumno> mostrarAlumnosConPractica();

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.service;

import com.franciscojosegarciacutillas.appinstituto.model.Alumno;
import com.franciscojosegarciacutillas.appinstituto.repository.IAlumnoRepository;
import java.util.ArrayList;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 *
 * @author Fran
 */
@Service
public class AlumnoServiceImpl implements IAlumnoService {

    @Autowired
    private IAlumnoRepository iRepository;

    @Override
    public Alumno registrar(Alumno alumno) {
        return iRepository.save( entity: alumno);
    }

    @Override
    public Alumno modificar(Alumno alumno) {
        return iRepository.save( entity: alumno);
    }

    @Override
    public ArrayList<Alumno> listar() {
        return (ArrayList<Alumno>) iRepository.findAll();
    }

    @Override
    public Alumno obtener(Integer id) {
        Optional<Alumno> op = iRepository.findById(id);
        return op.isPresent() ? op.get() : new Alumno();
    }

    @Override
    public void eliminar(Integer id) {
        iRepository.deleteById(id);
    }

    @Override
    public ArrayList<Alumno> mostrarAlumnosTitulo(String titulo) {
        return (ArrayList<Alumno>) iRepository.findByTitulo(titulo);
    }

    @Override
    public ArrayList<Alumno> mostrarAlumnosEdadMenorQue(Integer edad) {
        return (ArrayList<Alumno>) iRepository.findByEdadLessThan(edad);
    }

    @Override
    public ArrayList<Alumno> mostrarAlumnosSinPractica() {
        return (ArrayList<Alumno>) iRepository.findByPracticaNull();
    }

    @Override
    public ArrayList<Alumno> mostrarAlumnosConPractica() {
        return (ArrayList<Alumno>) iRepository.findByPracticaNotNull();
    }

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.service;

import com.franciscojosegarciacutillas.appinstituto.model.Empresa;
import java.util.ArrayList;

/**
 *
 * @author Fran
 */
public interface IEmpresaService {

    Empresa registrar(Empresa empresa);
    Empresa modificar(Empresa empresa);
    ArrayList<Empresa> listar();
    Empresa obtener(Integer id);
    void eliminar(Integer id);

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.service;

import com.franciscojosegarciacutillas.appinstituto.model.Empresa;
import com.franciscojosegarciacutillas.appinstituto.repository.IEmpresaRepository;
import java.util.ArrayList;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 *
 * @author Fran
 */
@Service
public class EmpresaServiceImpl implements IEmpresaService{

    @Autowired
    private IEmpresaRepository iRepository;

    @Override
    public Empresa registrar(Empresa empresa) {
        return iRepository.save( entity: empresa);
    }

    @Override
    public Empresa modificar(Empresa empresa) {
        return iRepository.save( entity: empresa);
    }

    @Override
    public ArrayList<Empresa> listar() {
        return (ArrayList<Empresa>)iRepository.findAll();
    }

    @Override
    public Empresa obtener(Integer id) {
        Optional<Empresa> op = iRepository.findById(id);
        return op.isPresent() ? op.get() : new Empresa();
    }

    @Override
    public void eliminar(Integer id) {
        iRepository.deleteById(id);
    }

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.service;

import com.franciscojosegarciacutillas.appinstituto.model.Practica;
import java.util.ArrayList;

/**
 *
 * @author Fran
 */
public interface IPracticaService {

    Practica registrar(Practica practica);
    Practica modificar(Practica practica);
    ArrayList<Practica> listar();
    Practica obtener(Integer id);
    void eliminar(Integer id);

}
```

```java
package com.franciscojosegarciacutillas.appinstituto.service;

import com.franciscojosegarciacutillas.appinstituto.model.Practica;
import com.franciscojosegarciacutillas.appinstituto.repository.IPracticaRepository;
import java.util.ArrayList;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 *
 * @author Fran
 */
@Service
public class PracticaServiceImpl implements IPracticaService{

    @Autowired
    private IPracticaRepository iRepository;

    @Override
    public Practica registrar(Practica practica) {
        return iRepository.save( entity: practica);
    }

    @Override
    public Practica modificar(Practica practica) {
        return iRepository.save( entity: practica);
    }

    @Override
    public ArrayList<Practica> listar() {
        return (ArrayList<Practica>)iRepository.findAll();
    }

    @Override
    public Practica obtener(Integer id) {
        Optional<Practica> op = iRepository.findById(id);
        return op.isPresent() ? op.get() : new Practica();
    }

    @Override
    public void eliminar(Integer id) {
        iRepository.deleteById(id);
    }

}
```

Dto

```java
package com.franciscojosegarciacutillas.appinstituto.dto;

import com.franciscojosegarciacutillas.appinstituto.model.Alumno;
import com.franciscojosegarciacutillas.appinstituto.model.Practica;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

/**
 *
 * @author Fran
 */
@Getter @Setter @AllArgsConstructor @NoArgsConstructor
public class AlumnoConPracticaDTO {

    private String nombre;
    private int edad;
    private String titulo;
    private Practica practica;

    public AlumnoConPracticaDTO(Alumno alumno){

        this.nombre = alumno.getNombreAlumno();
        this.edad = alumno.getEdad();
        this.titulo = alumno.getTitulo();
        this.practica = alumno.getPractica();

    }

}
```

Utilidades

```java
package com.franciscojosegarciacutillas.appinstituto.utilidades;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;


/**
 *
 * @author Fran
 */
@Getter @Setter @AllArgsConstructor
public class AlumnoPracticaUtil{

    private Integer alumnoId;
    private Integer practicaId;


}
```

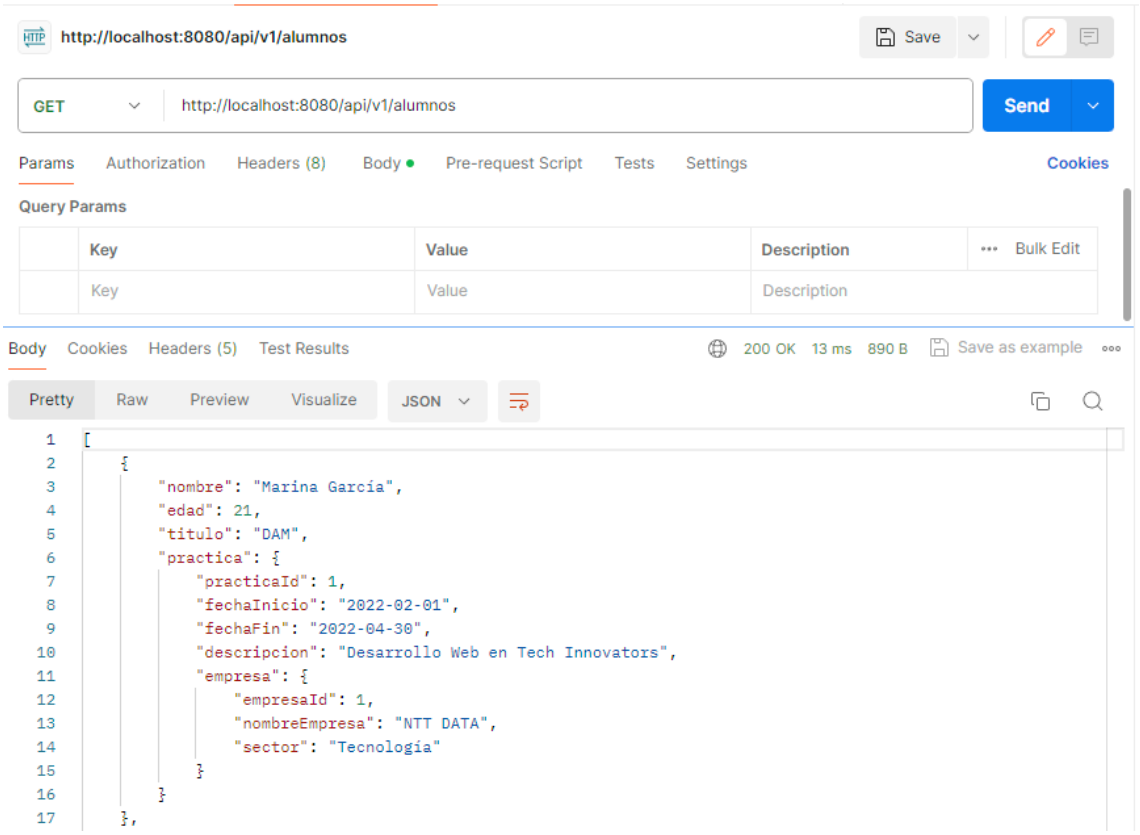
Controller

```java
package com.franciscojosegarciacutillas.appinstituto.controller;

import com.franciscojosegarciacutillas.appinstituto.dto.AlumnoConPracticaDTO;
import com.franciscojosegarciacutillas.appinstituto.model.Alumno;
import com.franciscojosegarciacutillas.appinstituto.model.Practica;
import com.franciscojosegarciacutillas.appinstituto.service.AlumnoServiceImpl;
import com.franciscojosegarciacutillas.appinstituto.service.PracticaServiceImpl;
import com.franciscojosegarciacutillas.appinstituto.utilidades.AlumnoPracticaUtil;
import java.util.ArrayList;
import java.util.Objects;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 *
 * @author Fran
 */
@RestController
@RequestMapping("/api/v1")
public class AlumnoController {

    @Autowired
    AlumnoServiceImpl aluServ;

    @Autowired
    PracticaServiceImpl pracServ;
```

Tomando como referencia dicha base de datos deberás generar un API Rest que proporcione los siguientes endpoints

1. Consultar todos los alumnos con su práctica (si la tuviera asignada)

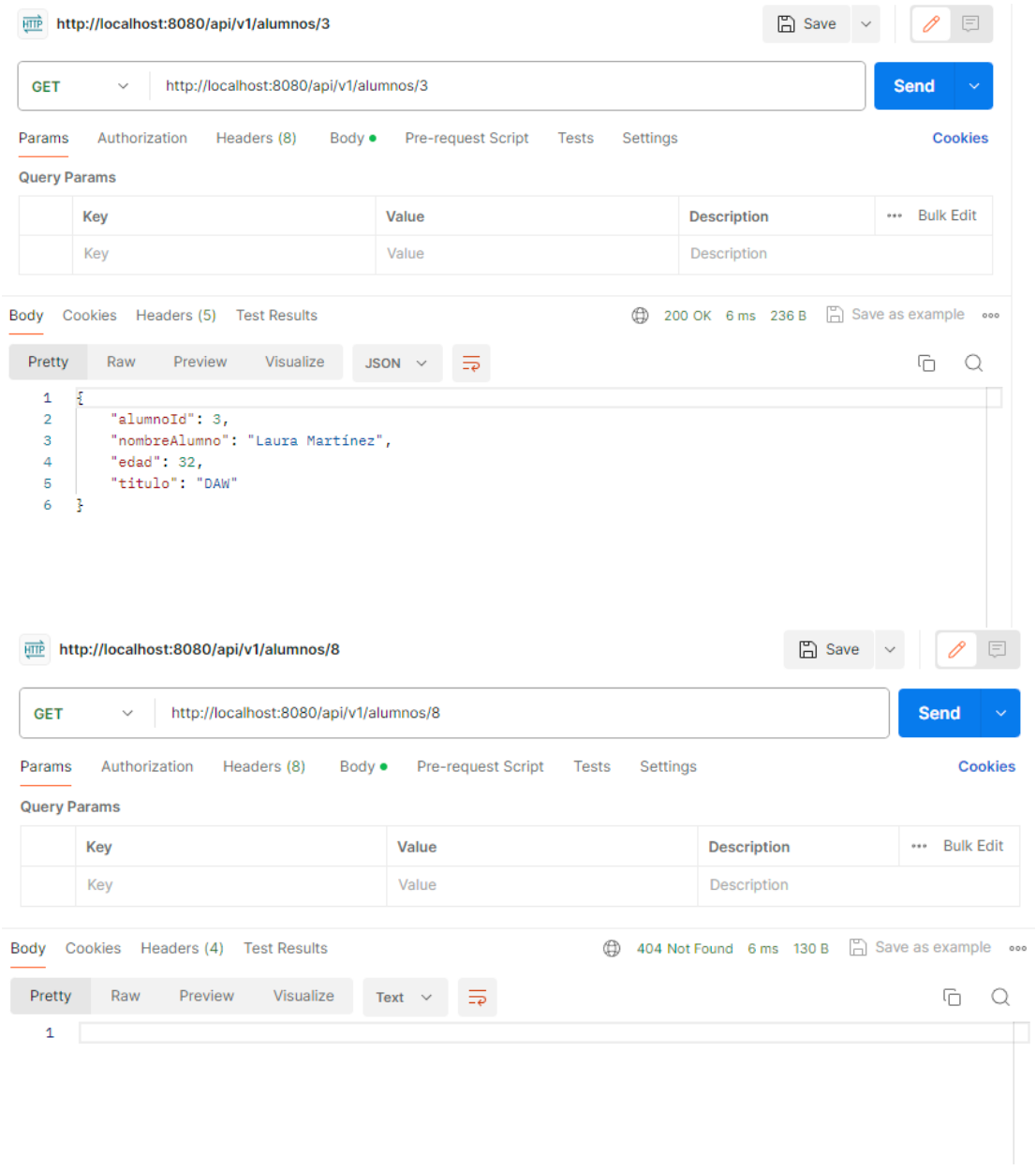
```java
//Recuperar todos los alumnos
@GetMapping("/alumnos")
public ResponseEntity<ArrayList<AlumnoConPracticaDTO>> getAlumnosConPractica() {

    ArrayList<Alumno> alumnos = aluServ.listar();
    ArrayList<AlumnoConPracticaDTO> alumnosDevolver = new ArrayList<>();

    for (Alumno alumno : alumnos) {

        AlumnoConPracticaDTO nuevoAlumno = new AlumnoConPracticaDTO(alumno);
        alumnosDevolver.add( e: nuevoAlumno);

    }

    return new ResponseEntity<>( body: alumnosDevolver,  status: HttpStatus.OK);

}
```




```
[
    {
        "nombre": "Marina García",
        "edad": 21,
        "titulo": "DAM",
        "practica": {
            "practicaId": 1,
            "fechaInicio": "2022-02-01",
            "fechaFin": "2022-04-30",
            "descripcion": "Desarrollo Web en Tech Innovators",
            "empresa": {
                "empresaId": 1,
                "nombreEmpresa": "NTT DATA",
                "sector": "Tecnología"
            }
        }
    },
    {
        "nombre": "Alejandro Ruiz",
        "edad": 43,
        "titulo": "DAW",
        "practica": {
            "practicaId": 2,
            "fechaInicio": "2022-03-15",
            "fechaFin": "2022-06-15",
            "descripcion": "Análisis de Datos en Data Wizards",
            "empresa": {
                "empresaId": 2,
                "nombreEmpresa": "NEORIS",
                "sector": "Data Science"
            }
        }
    },
    {
        "nombre": "Laura Martínez",
        "edad": 32,
        "titulo": "DAW",
        "practica": null
    },
    {
        "nombre": "Javier López",
        "edad": 24,
        "titulo": "DAM",
        "practica": null
    },
    {
        "nombre": "Sara Rodríguez",
        "edad": 21,
        "titulo": "ASIR",
        "practica": null
    }
]
```

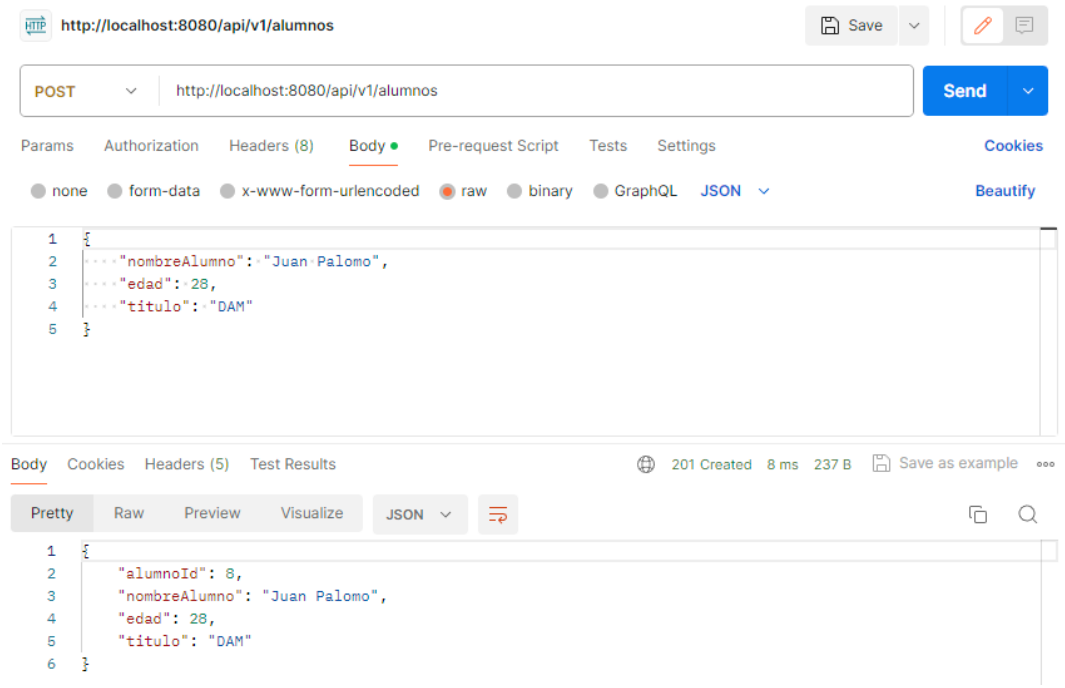
2. Consultar los datos de un alumno concreto

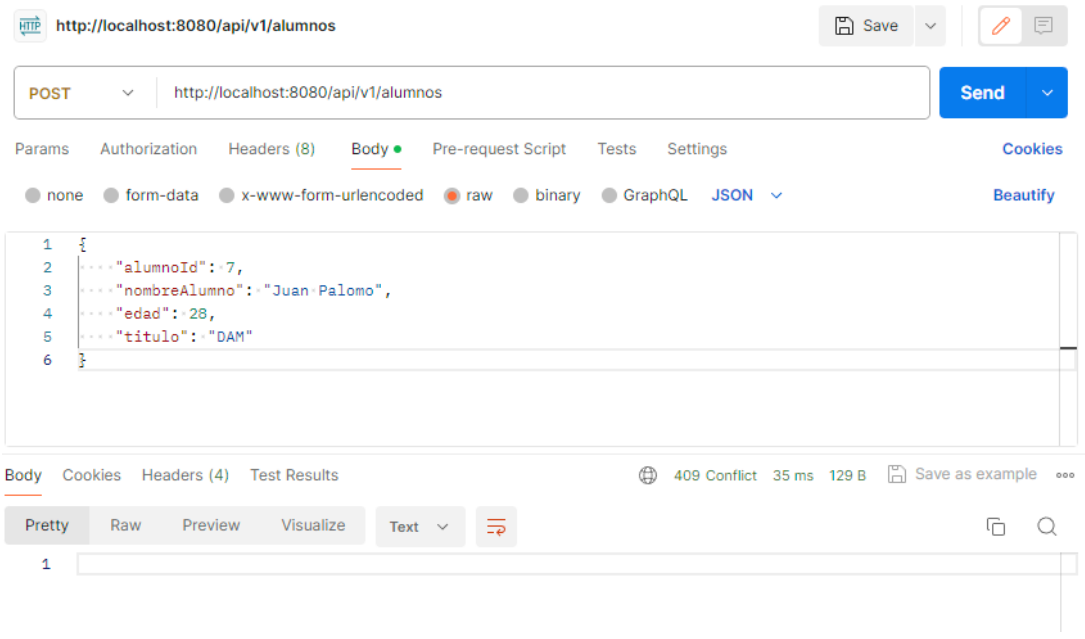
```java
//Recuperar los datos de un alumno en concreto
@GetMapping("/alumnos/{id}")
public ResponseEntity<Alumno> getAlumnoId(@PathVariable Integer id) {

    Alumno alumno = aluServ.obtener(id);

    if (alumno.getAlumnoId() == null) {

        return new ResponseEntity<>( status:HttpStatus.NOT_FOUND);

    } else {

        return new ResponseEntity<>( body: alumno,  status:HttpStatus.OK);

    }

}
```



3. Insertar un alumno (sin práctica)

```java
//Insertar un alumno sin práctica
@PostMapping("/alumnos")
public ResponseEntity<Alumno> postAlumnoSinPractica(@RequestBody Alumno alumno) {

    if (alumno.getAlumnoId()!=null){
        Alumno alumnoExiste = aluServ.obtener( id: alumno.getAlumnoId());
        if (alumnoExiste.getAlumnoId()!=null){
            return new ResponseEntity<>( status: HttpStatus.CONFLICT);
        }
    }

    Alumno alumnoInsertar = aluServ.registrar(alumno);

    return new ResponseEntity<>( body: alumnoInsertar,  status: HttpStatus.CREATED);

}
```

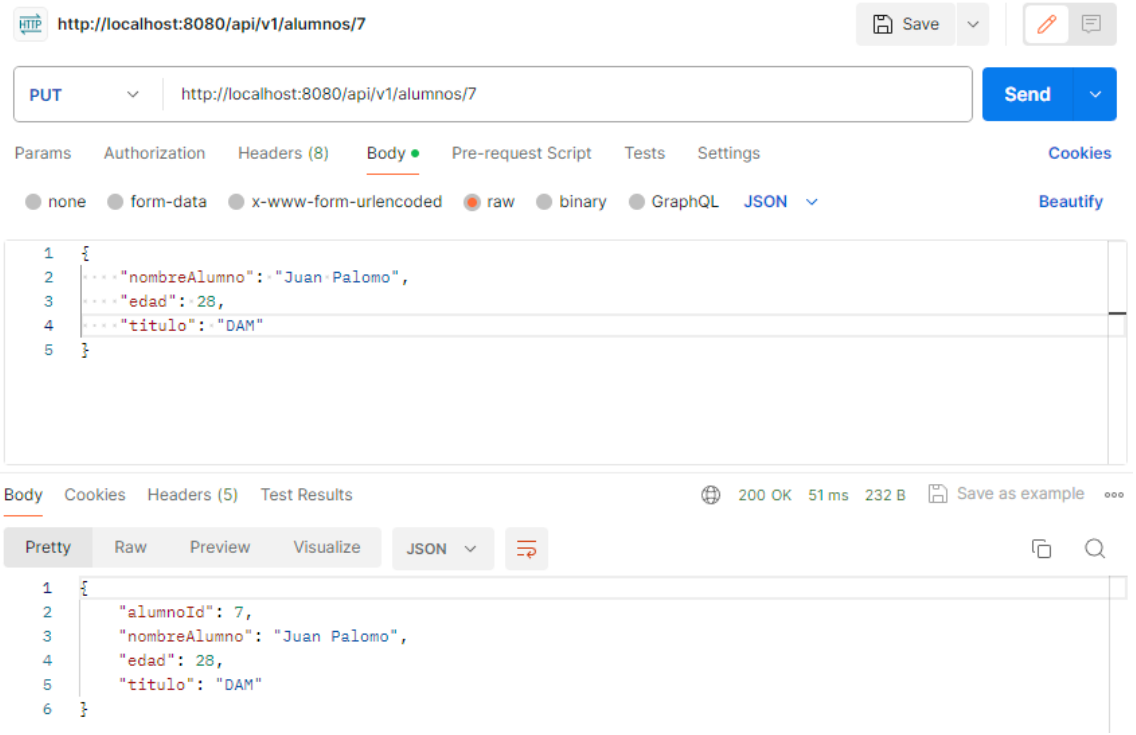




4. Actualizar datos de un alumno

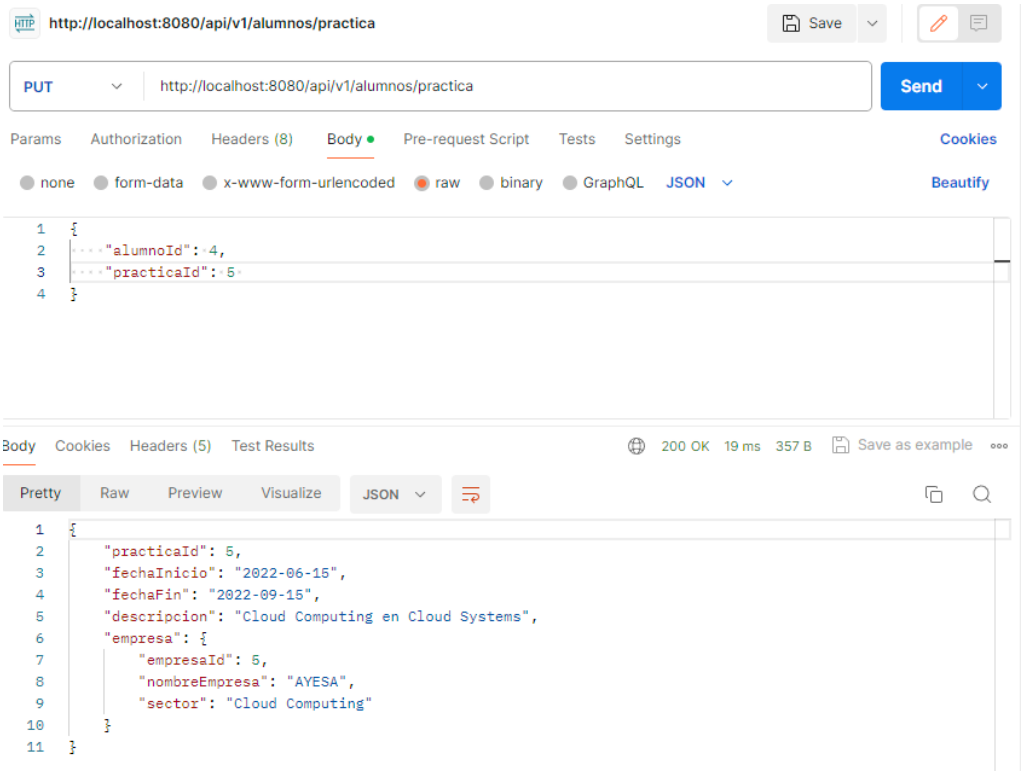
```java
//Actualizar datos de un alumno
@PutMapping("/alumnos/{id}")
public ResponseEntity<Alumno> putAlumno(@RequestBody Alumno alumno, @PathVariable Integer id) {

    Alumno alumnoModificar = aluServ.obtener(id);

    if (alumnoModificar.getAlumnoId() == null) {

        return new ResponseEntity<>( status: HttpStatus.NOT_FOUND);

    } else {
        if (!Objects.equals( a: alumno.getAlumnoId(), b: id) && alumno.getAlumnoId() != null) {

            return new ResponseEntity<>( status: HttpStatus.BAD_REQUEST);

        } else {

            alumnoModificar.setNombreAlumno( nombreAlumno: alumno.getNombreAlumno());
            alumnoModificar.setEdad( edad: alumno.getEdad());
            alumnoModificar.setTitulo( titulo: alumno.getTitulo());

            alumnoModificar = aluServ.modificar( alumno: alumnoModificar);

        }
    }

    return new ResponseEntity<>( body: alumnoModificar,  status: HttpStatus.OK);

}
```

Le cambiamos de ASIR a DAM.

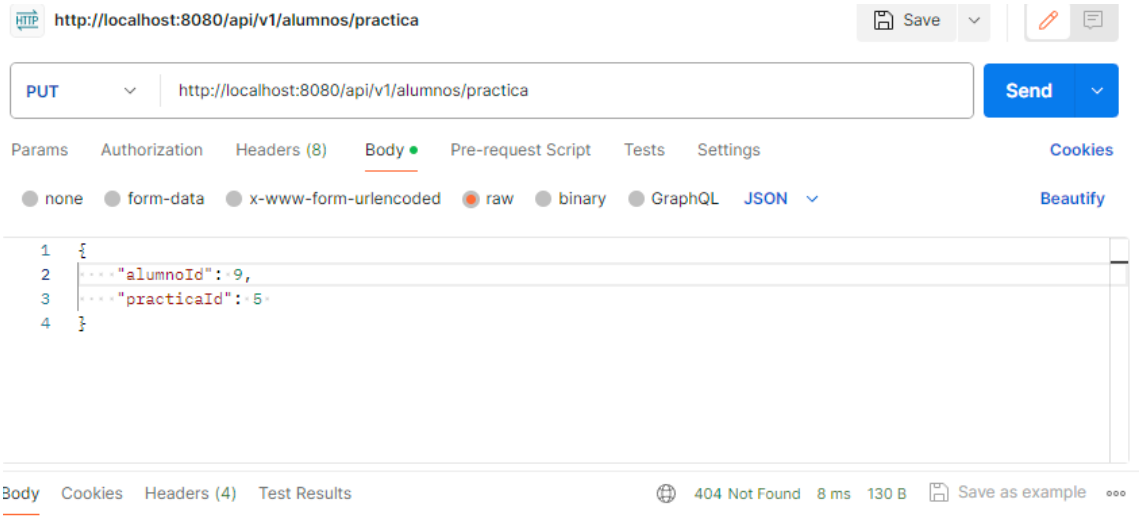


5. Insertar una práctica para un alumno

```java
//Insertar una práctica para un alumno
@PutMapping("/alumnos/practica")
public ResponseEntity<Practica> putPracticaAlumno(@RequestBody AlumnoPracticaUtil alumnoPracticaUtil){

    Practica practicaInsertar = pracServ.obtener( id: alumnoPracticaUtil.getPracticaId());
    Alumno alumnoPractica = aluServ.obtener( id: alumnoPracticaUtil.getAlumnoId());

    if (practicaInsertar.getPracticaId() == null || alumnoPractica.getAlumnoId() == null) {

        return new ResponseEntity<>( status: HttpStatus.NOT_FOUND);

    }

    practicaInsertar.setAlumnoId( alumnoId: alumnoPractica);
    practicaInsertar = pracServ.modificar( practica: practicaInsertar);

    return new ResponseEntity<>( body: practicaInsertar,  status: HttpStatus.OK);

}
```

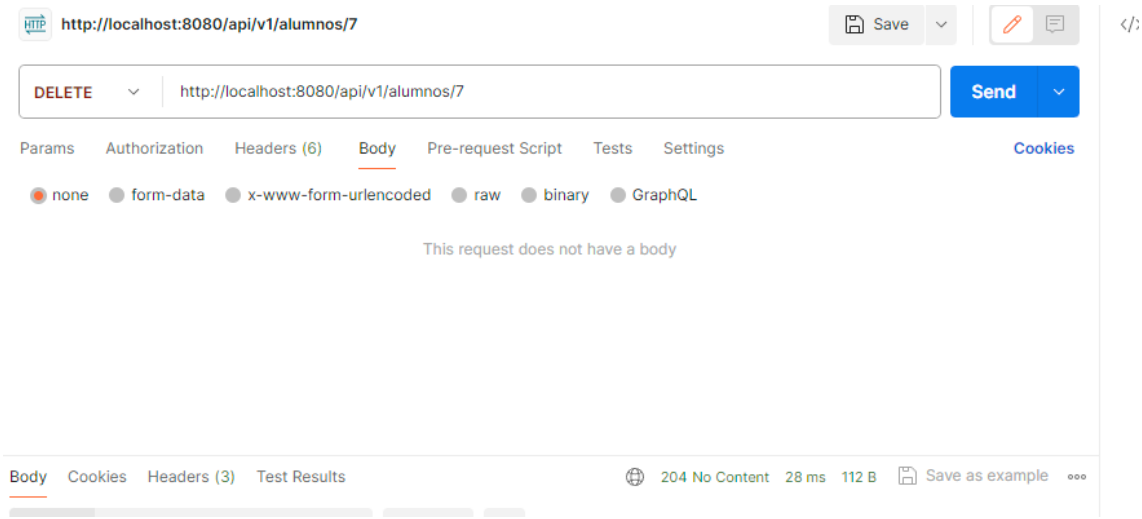


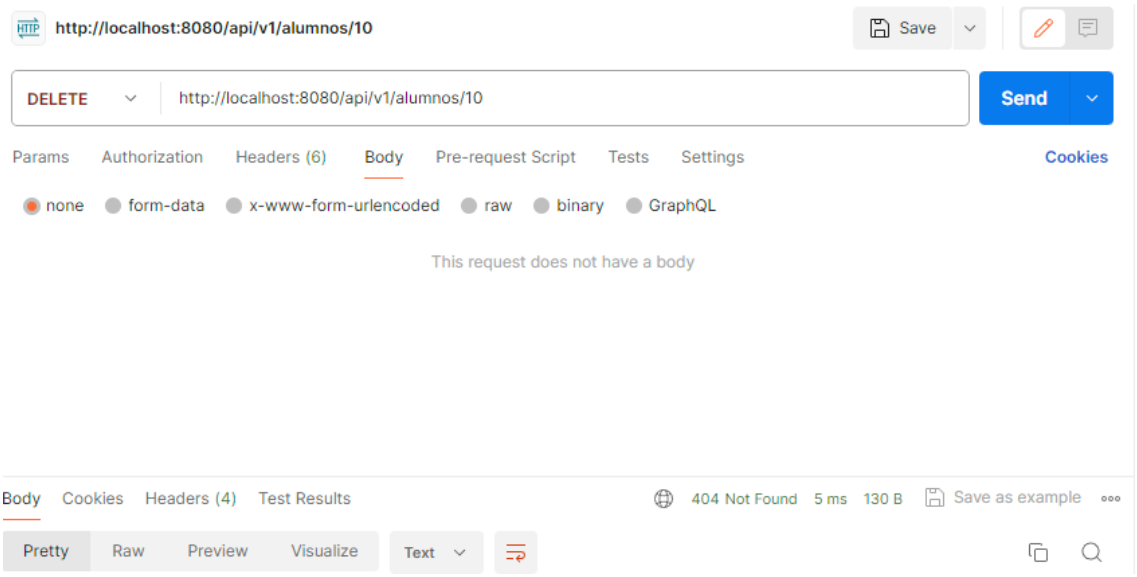

```json
{
    "nombre": "Javier López",
    "edad": 24,
    "titulo": "DAM",
    "practica": {
        "practicaId": 5,
        "fechaInicio": "2022-06-15",
        "fechaFin": "2022-09-15",
        "descripcion": "Cloud Computing en Cloud Systems",
        "empresa": {
            "empresaId": 5,
            "nombreEmpresa": "AYESA",
            "sector": "Cloud Computing"
        }
    }
}
```



6. Eliminar un alumno.

```
//Eliminar un alumno
@DeleteMapping("/alumnos/{id}")
public ResponseEntity<Alumno> deleteAlumno(@PathVariable Integer id) {

    Alumno alumno = aluServ.obtener(id);
    if (alumno.getAlumnoId()== null) {

        return new ResponseEntity<>( status:HttpStatus.NOT_FOUND);

    }

    aluServ.eliminar(id);
    return new ResponseEntity<>( status:HttpStatus.NO_CONTENT);

}
```

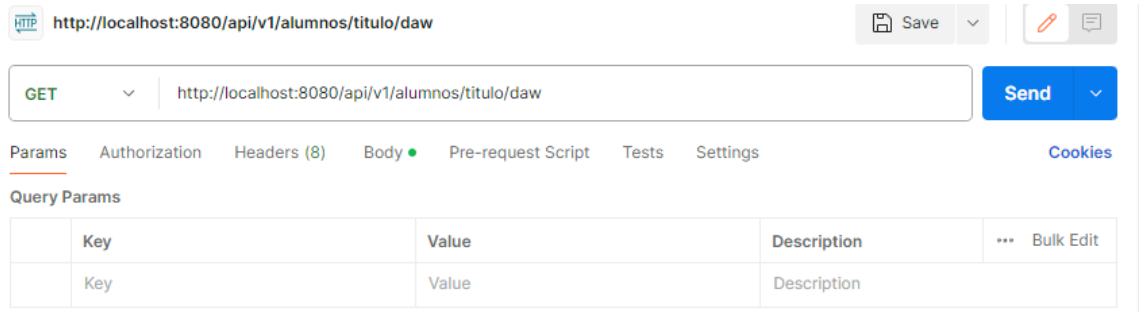


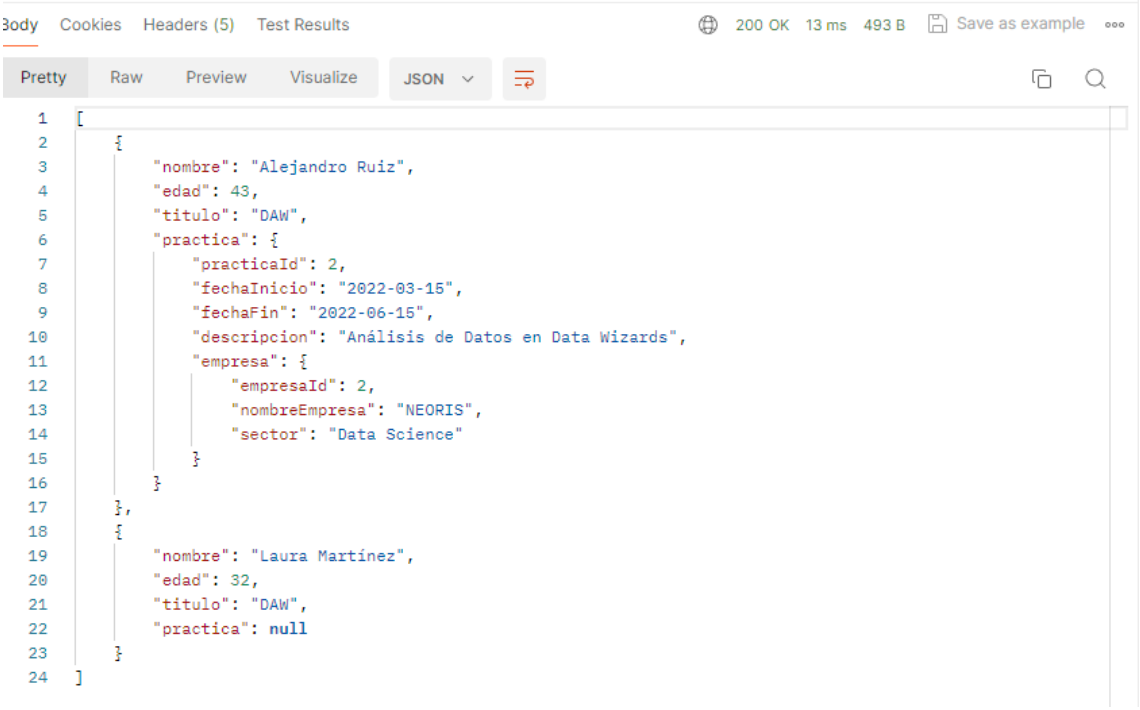


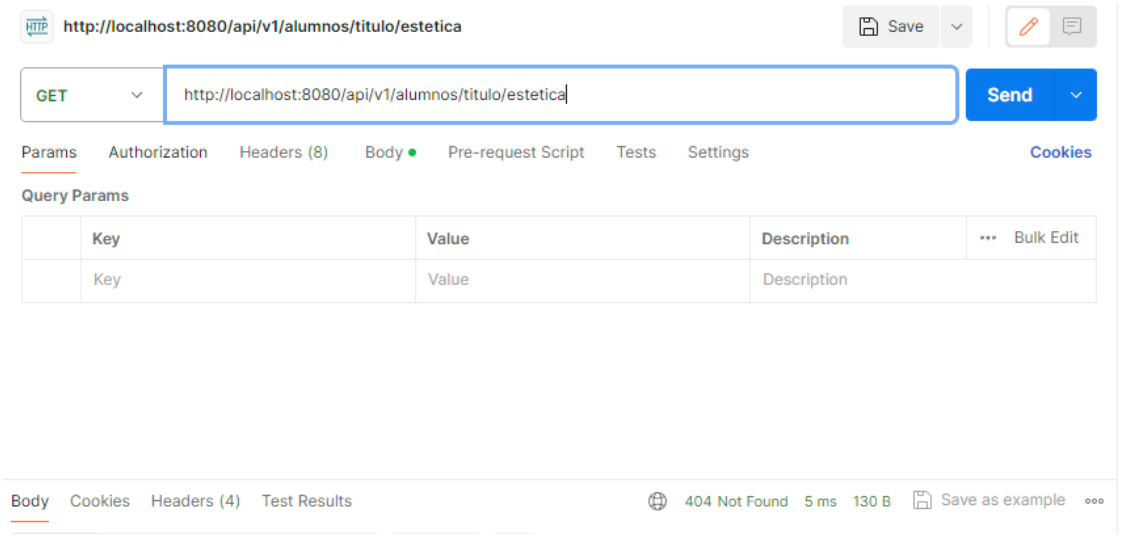
Haciendo uso de las consultas automatizadas de Spring Data

7. Devolver todos los alumnos con su práctica (si la tuviera asignada) de un determinado título

```
//Devolver todos los alumnos con su práctica de un determinado título
@GetMapping("/alumnos/titulo/{titulo}")
public ResponseEntity<ArrayList<AlumnoConPracticaDTO>> findByTitulo(@PathVariable String titulo){

    ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosTitulo(titulo);
    ArrayList<AlumnoConPracticaDTO> alumnosDevolver = new ArrayList<>();

    if (alumnos.isEmpty()) {

        return new ResponseEntity<>( status:HttpStatus.NOT_FOUND);

    }

    for (Alumno alumno : alumnos) {

        AlumnoConPracticaDTO nuevoAlumno = new AlumnoConPracticaDTO(alumno);
        alumnosDevolver.add( e:nuevoAlumno);

    }

    return new ResponseEntity<>( body:alumnosDevolver,  status:HttpStatus.OK);
}
```

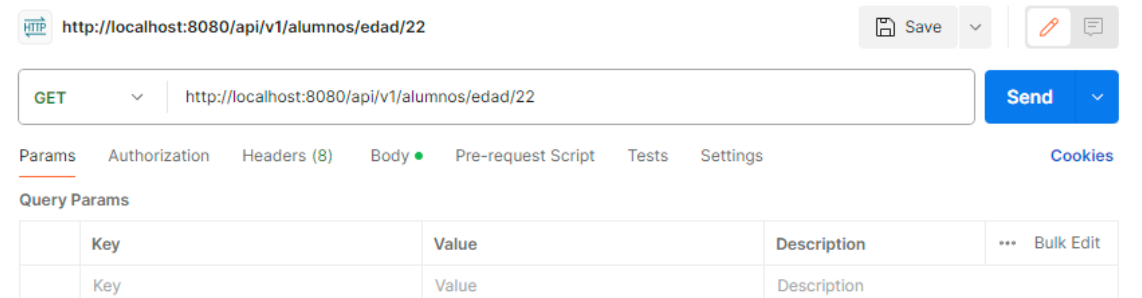

http://localhost:8080/api/v1/alumnos/titulo/daw
Save
GET
http://localhost:8080/api/v1/alumnos/titulo/daw
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description

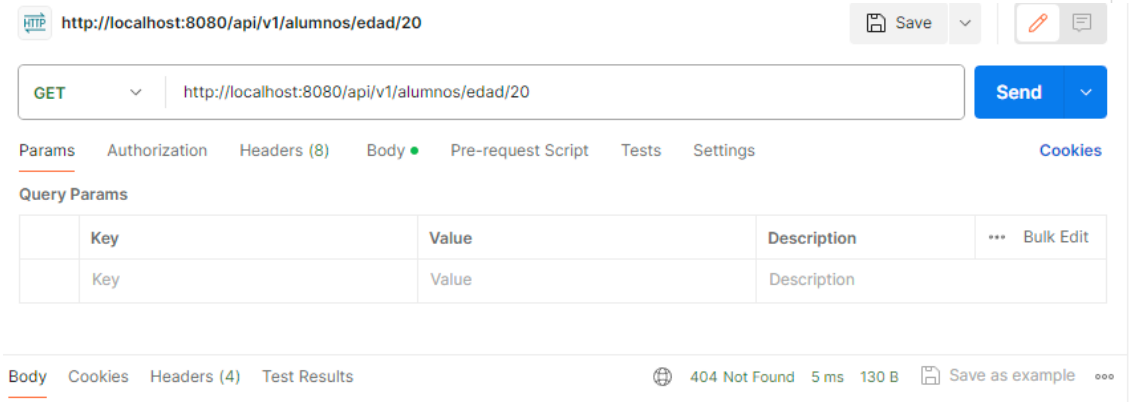

Body
Cookies
Headers (5)
Test Results
200 OK
13 ms
493 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"nombre": "Alejandro Ruiz",
"edad": 43,
"titulo": "DAW",
"practica": {
"practicaId": 2,
"fechaInicio": "2022-03-15",
"fechaFin": "2022-06-15",
"descripcion": "Análisis de Datos en Data Wizards",
"empresa": {
"empresaId": 2,
"nombreEmpresa": "NEORIS",
"sector": "Data Science"
}
}
},
{
"nombre": "Laura Martínez",
"edad": 32,
"titulo": "DAW",
"practica": null
}
]


http://localhost:8080/api/v1/alumnos/titulo/estetica
Save
GET
http://localhost:8080/api/v1/alumnos/titulo/estetica
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
404 Not Found
5 ms
130 B
Save as example

8. Devolver todos los alumnos que tienen menos de cierta edad

```java
//8.Devolver todos los alumnos que tienen menos de cierta edad
@GetMapping("/alumnos/edad/{edad}")
public ResponseEntity<ArrayList<Alumno>> getAlumnoEdadMenorQue(@PathVariable Integer edad){

    ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosEdadMenorQue(edad);

    if (alumnos.isEmpty()) {

        return new ResponseEntity<>( status: HttpStatus.NOT_FOUND);

    }

    return new ResponseEntity<>( body: alumnos,  status: HttpStatus.OK);

}
```

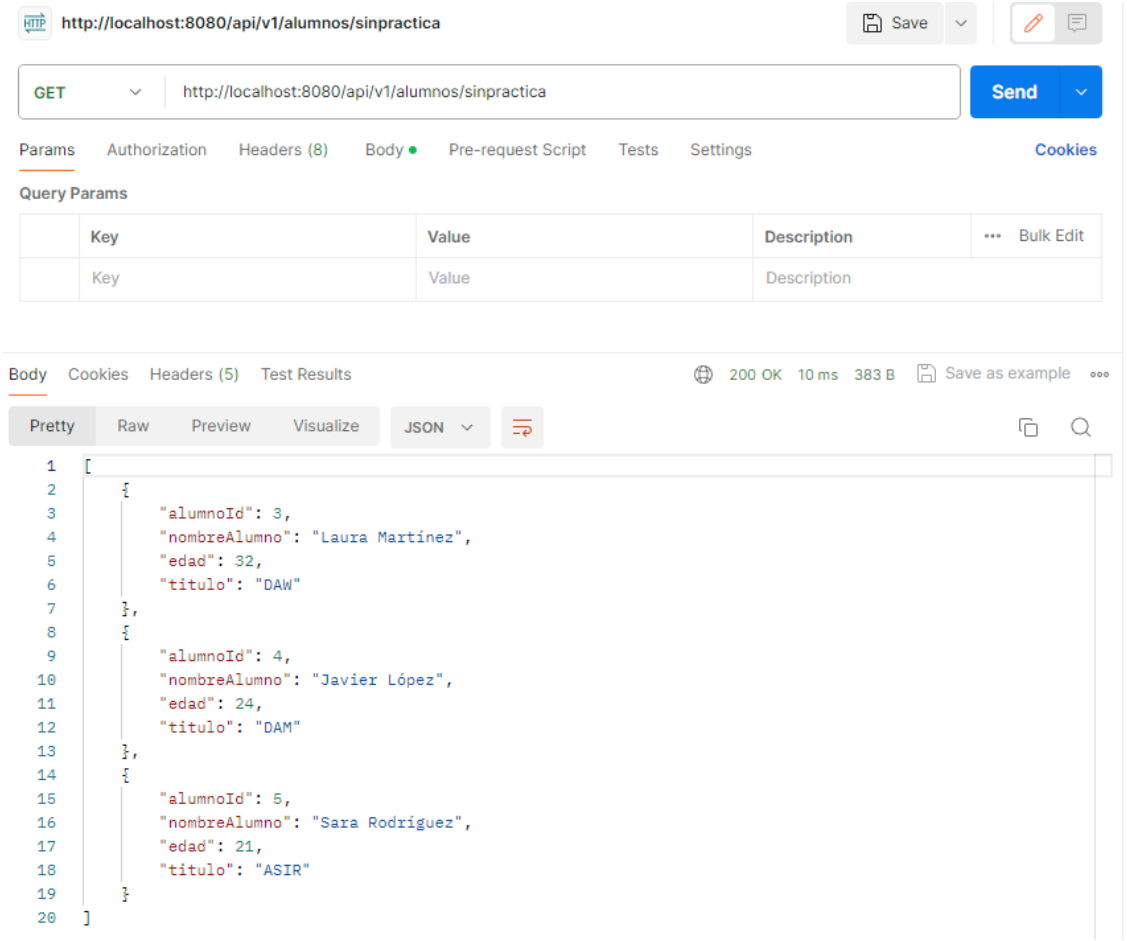




Haciendo uso de consultas JPQL

9. Obtener todos los alumnos que no tienen práctica

```java
//9.Obtener todos los alumnos que no tienen práctica
@GetMapping("/alumnos/sinpractica")
public ResponseEntity<ArrayList<Alumno>> getAlumnosSinPractica(){

    ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosSinPractica();

    if (alumnos.isEmpty()) {

        return new ResponseEntity<>( status: HttpStatus.NOT_FOUND);

    }

    return new ResponseEntity<>( body: alumnos,  status: HttpStatus.OK);

}
```



10. Obtener todos los alumnos que tienen práctica asignada.

```java
//10.Obtener todos los alumnos que tienen práctica asignada
@GetMapping("/alumnos/conpractica")
public ResponseEntity<ArrayList<Alumno>> getAlumnosConPracticaAsignada(){

    ArrayList<Alumno> alumnos = aluServ.mostrarAlumnosConPractica();

    if (alumnos.isEmpty()) {

        return new ResponseEntity<>( status: HttpStatus.NOT_FOUND);

    }

    return new ResponseEntity<>( body: alumnos,  status: HttpStatus.OK);

}
```