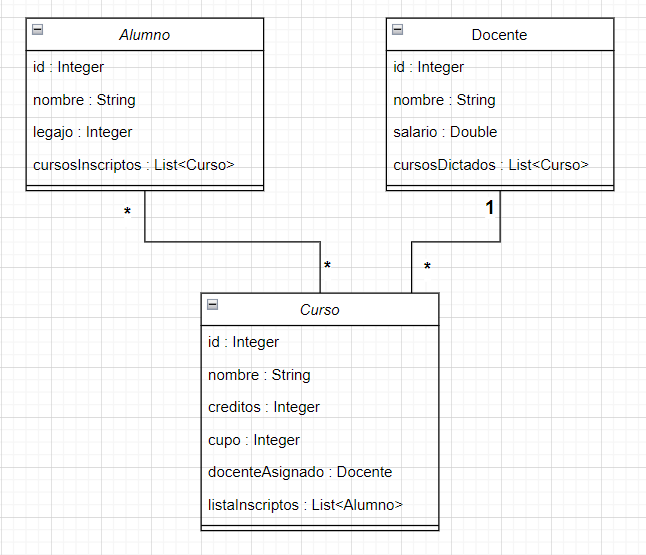
**Desarrollo de Aplicaciones en la Nube**

**Práctica de Spring Framework**

|  |
| --- |
| **Autoayuda**  Inicializar un proyecto de Java 21, Maven y Spring Boot con las siguientes dependencias:   * Spring Web   Escriba en el grupo de metadatos:   * Group: isi.dan.practicas * Artifact: practica<N> * Name (no modificar): practica<N> * Description: Inicializo la práctica <N> * Package name (no modificar): isi.dan.practicas.practica<N>   Reemplazar <N> por el Nº de la práctica que se esté desarrollando |

**Práctica 1:**

Dada la siguiente imagen:



1. Desarrolle los modelos dentro de la carpeta “model”. Con los atributos privados, sus propios constructores, getters/setters.
2. Desarrolle las excepciones CupoExcedidoException, DocenteExcedidoException y RecursoNoEncontrado dentro de la carpeta “exception” y que extienda de la clase Exception. Donde el CupoExcedidoException debe mostrar un mensaje de error que el curso alcanzó el cupo máximo para alumnos inscriptos, DocenteExcedidoException donde el docente ya tiene 3 cursos asignados y no puede asignar más cursos, y finalmente RecursoNoEncontrado va a tener 2 parámetros, el primero corresponde una cadena que puede ser “Alumno”, “Docente” o “Curso” y el segundo corresponde al identificador, el mensaje final debe ser “No existe el identificador 7 del modelo Cliente”.
3. Resolver las siguientes consignas:
   1. Un método que retorne la cantidad de que un docente tiene asignado.
   2. Un docente puede ser asignado a 3 cursos como máximo. Desarrolle en la clase Curso y lanzar la excepción DocenteExcedidoException en caso que se exceda.
   3. Un método que permita agregar un curso a la lista de cursos dictados por el docente
   4. Inscribir un alumno a un curso, que no exceda del cupo del curso. En caso que se exceda, lanzar la excepción de CupoExcedidoException.
4. Generar interfaces de servicio para cada modelo con los siguientes métodos, dentro de la carpeta “service”:

* public [T] guardar[T]([T] a);
* public Optional<[T]> buscar[T]PorId(Integer id);
* public List<[T]> listar[T]s();
* public void baja[T](Integer id);

Nota: reemplazar [T] por el nombre del modelo.

Generar los métodos que consideres necesarios para trabajar con los métodos generados en el inciso ‘c’

1. Generar las implementaciones de los servicios que implemente las interfaces generadas en el inciso anterior.

* Curso necesita enlazar con los servicios de Alumno y Docente ya que va a trabajar con inscribirAlumno y asignarDocente.
* Cada una de ella, inicializar un atributo de contador de ID estático y una lista.
* Para los métodos de guardar[T]
  + Si el modelo pasado por parámetro no tiene id, guardar como nuevo a la lista.
  + Si el modelo pasado por parámetro tiene id, puede suceder 2 formas:
    - Existe en la lista, reemplazar a ella.
    - No existe en la lista, lanzar una excepción de RecursoNoEncontrado.
* Para los métodos de baja[T]
  + Si el identificador pasado por parámetro no existe en la lista, lanzar una excepción de RecursoNoEncontrado.
  + Si el identificador existe, eliminarlo de la lista.
* Para los métodos de listar, retornar la lista
* Para los métodos de buscar[T], si se encuentra retornarlo, sino lanzar una excepción de RecursoNoEncontrado.
* Trabajar con el resto de los métodos que se trabajó en el inciso ‘c’.

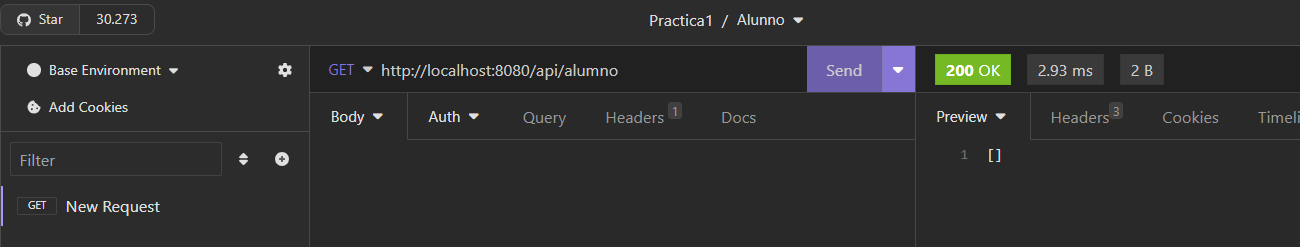
1. Generar los controladores de cada modelo dentro de la carpeta “controller”, donde trabaje con REST.
2. Puede trabajar con la aplicación con las siguientes herramientas
   1. ThunderClient (VSCode Plugin <https://marketplace.visualstudio.com/items?itemName=rangav.vscode-thunder-client> )
   2. Insomnia (<https://insomnia.rest/download>)
   3. Postman (<https://www.postman.com/downloads/>) para probar todos los métodos de los controladores generado en el inciso ‘f’:

Realizar las siguientes pruebas

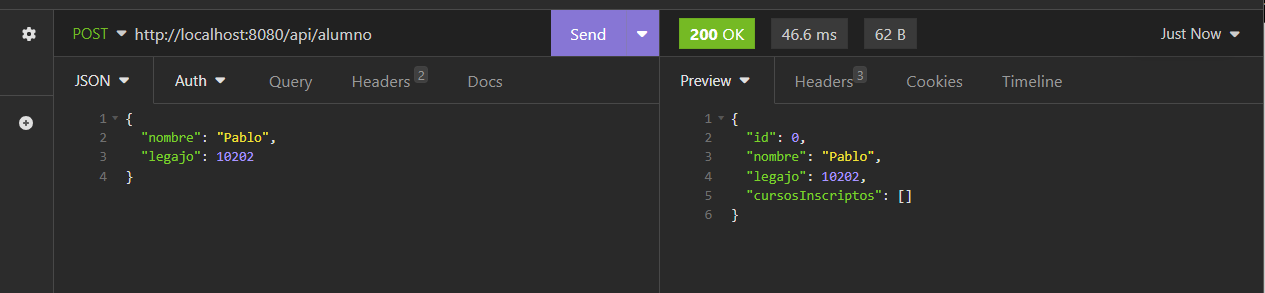
* Crear un alumno, editarlo, listar y luego borrarlo.
* Crear un docente, editarlo, listar y luego borrarlo.
* Crear un curso, editarlo, listar y luego borrarlo.
* Crear 1 docente, 5 cursos, tratar de asignar ese único docente a los 5 y verificar que esté la excepción lanzada
* Crear 1 curso con cupo máximo de 5 alumnos, inscribir 6 alumnos y verificar que esté la excepción lanzada.

Ejemplo con Insomnia:

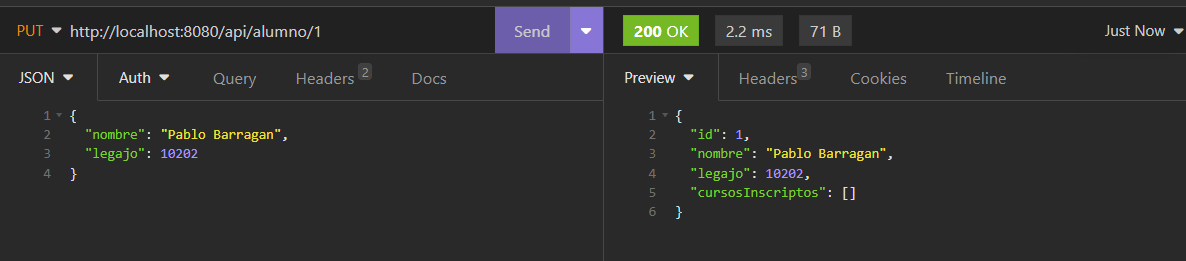
Listar alumnos (inicialmente vacía)



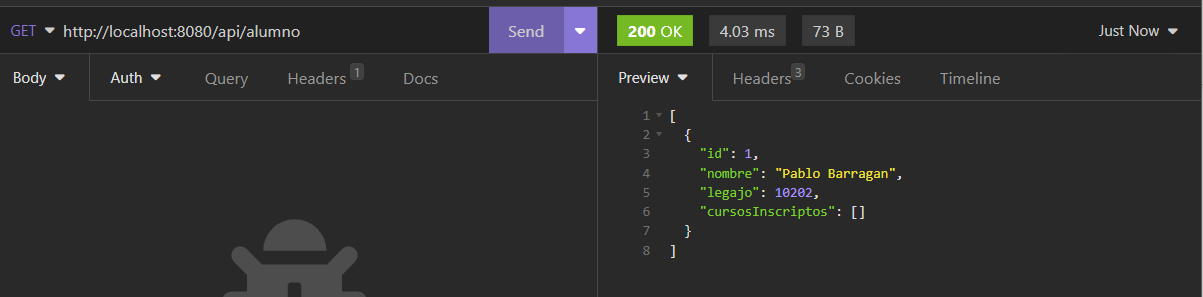
Crear alumno



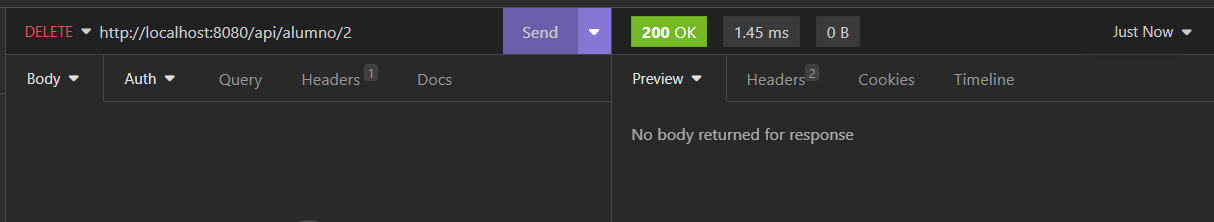
Actualizar Alumno



Listar alumnos

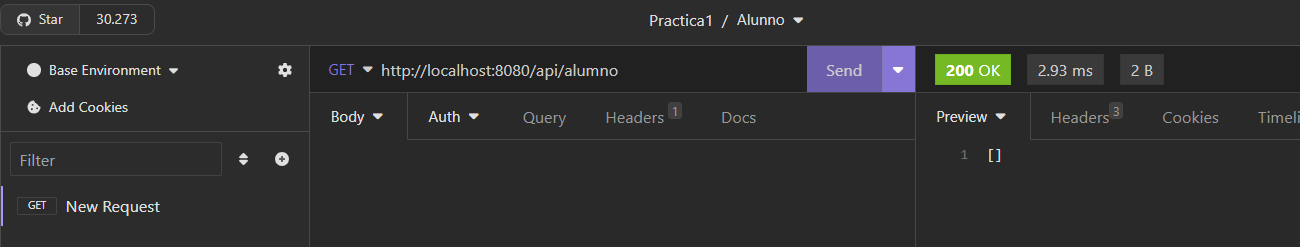


Baja alumno



Nota: no hay respuesta en el cuerpo, pero sí en el código (200 OK)

Listar alumnos



**Práctica 1.1 Trabajar con GIT**

Descargar GIT (<https://git-scm.com/downloads>)

Crear una cuenta o iniciar sesión en GitHub (<https://github.com/>)

Crear un nuevo repositorio con el nombre “dan-practica01”, público y sin nada extras.

En el nuevo repositorio se van a encontrar el siguiente cuadro (siempre que no hayan incluido el Readme.md para poder visualizarlo):



Va a ser de ayuda para entender bien que hay que hacer, sin embargo, no vamos a ejecutarlos así, sino de la siguiente manera:

|  |
| --- |
| git init  git add . ß (atento, se le puso un punto después del add)  git commit -m “first commit”  git branch -M main  git remote add origin <MiUrlDelRepositorioNuevo>  git push -u origin main |

