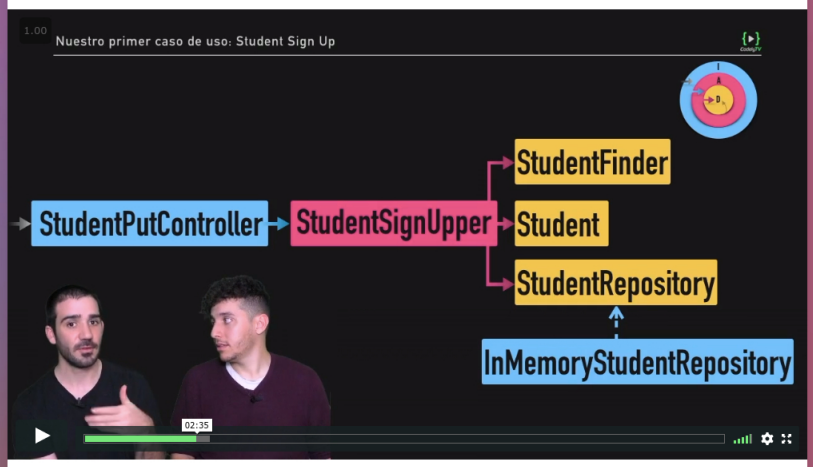
Watch: **Nuestro primer caso de uso: Student Sign Up**
Completed · 15m · 100 Points



Caso de Uso

Nuestro caso de uso consistirá en el registro de un nuevo estudiante en la plataforma, y el flujo que idealmente llevará es el siguiente:

1. Llegará una request al Controlador `StudentPutController`
2. Desde el Controlador se llamará al Servicio de Aplicación `StudentSignUpper`. Este servicio interactuará con
   - El Servicio de Dominio `StudentFinder` para comprobar si ya existe ese usuario
   - El Agregado `Student`, entendido en este caso como una Entidad
   - El repositorio `StudentRepository` donde se almacenará el estudiante

Es interesante señalar cómo desde el propio nombre del controlador vemos que está acoplándose al protocolo HTTP, esto es así precisamente porque es el punto en el que conocemos el protocolo de comunicación con el cliente (Estamos conociendo en este caso que es una API HTTP, y que recibiremos y enviaremos un json). Además, esta manera de nombrar al controlador nos obligará a que sólo tenga un endpoint y pertenezca a un único caso de uso.

Podéis ver aquí el repo de este ejemplo

Feature `Student_Put`:

```
Feature: Sign up a student
  In order to learn from CodelyTV Pro courses
  As an anonymous student
  I want to sign up to the platform

  Scenario: Sign up a new student
    Given I send a PUT request to "/students/0ca24fc4-bdc8-48d0-9c5f-94183a627adc" with
body:
    """
    {
      "name": "javi",
      "password": "superSecret"
    }
    """
    Then the response status code should be 201
    And the response should be empty
```

Partimos de este test en el que sencillamente estaríamos especificando que dado que enviamos una petición al controlador con un nuevo estudiante, esperamos que nos responda con un código de estado 201, indicando que todo ha ido bien 👍

Clase `StudentPutController`:

```
final class StudentPutController extends ApiController
{
    private $signUpper;

    public function __construct(StudentSignUpper $signUpper)
    {
        $this->signUpper = $signUpper;
    }

    public function __invoke(string $id, Request $request)
    {
        $this->signUpper->__invoke($id, $request->get('name'),$request-
>get('password'));

        return new ApiHttpCreatedResponse();
    }
}
```

Como veíamos en el flujo de la petición, el `StudentPutController` llamará al application service `StudentSignUpper`, que vamos a crear en la carpeta Application dentro del módulo de Student. Este application service lo estaríamos inyectando en el constructor del controller.

Encapsular la lógica de negocio en el application service nos permitirá que podamos aprovecharla desde múltiples puntos de entrada.

Clase `StudentSignUpper`:

```
final class StudentSignUpper
{
```

```
    private static $students = [];

    public function __invoke(string $studentId, string $studentName, string
$studentPassword){
        $existentStudent = get($studentId, self::$students);

        if(null !== $existentStudent)
        {
            throw new StudentAlreadyExist($studentId)
        }

        $student = new Student($studentId,$studentName,$studentPassword);

        self::$students[$studentId] = $studentId;
    }
}
```

Dentro del *invoke* de este servicio estaríamos comprobando si el estudiante ya existe, lanzando en ese caso una excepción `StudentAlreadyExist`. Ojo 👀! Se trata de una excepción de dominio, por lo que la crearemos dentro de la carpeta Domain junto a las demás clases de dominio de este módulo; con esto estaremos ganando en semántica dentro de nuestra aplicación.

En caso de que no exista, simplemente lo persistiríamos en base de datos o, en nuestro caso de ejemplo, en memoria. Vemos como una vez registrado el nuevo estudiante no estamos devolviendo nada, en nuestra opinión las mutaciones del estado global de nuestra aplicación deberían hacerse sin retornar nada por temas de separación de responsabilidades. Si queréis profundizar más en este tema, podéis hacerlo en el curso de CQRS

¿Alguna Duda?

Ya hemos visto cómo sería el flujo de nuestro caso de uso "registrar un nuevo estudiante", esperamos que os haya ayudado! Nos vemos en la siguiente lección 👉👉 Value Objects: Modelando nuestro dominio ⚛️

Si tienes dudas o sugerencias sobre esta lección recuerda que puedes abrir una nueva discusión ¡Justo aquí abajo!

👇👇👇