### [Microservicios con Spring Boot, Cloud y Docker](https://www.udemy.com/course/microservicios-spring-boot-cloud/)

**15. Desarrollo de un servicio completo**

Se hace una copia de 02\_microservicio\_cursos, se agregan los métodos PUT ,POST, DELETE

@DeleteMapping(value="curso/{name}")

public void eliminarCurso(@PathVariable("name") String nombre) {

cursos.removeIf(c->c.getNombre().equals(nombre));

}

@PostMapping(value="curso",consumes=MediaType.APPLICATION\_JSON\_VALUE,produces=MediaType.APPLICATION\_JSON\_VALUE)

public List<Curso> altaCurso(@RequestBody Curso curso){

cursos.add(curso);

return cursos;

}

@PutMapping(value="curso",consumes=MediaType.APPLICATION\_JSON\_VALUE,produces=MediaType.APPLICATION\_JSON\_VALUE)

public List<Curso> actualizaCurso(@RequestBody Curso curso){

for(int i=0; i<cursos.size(); i++) {

if(cursos.get(i).getNombre().equals(curso.getNombre())) {

cursos.set(i, curso);

}

}

return cursos;

}

<http://localhost:8080/cursos> muestra cursos

<http://localhost:8080/curso/Phyton> elimina un curso

**16. Testing de microservicios**

Nota: Como es complicado probar cada método desde el navegador, se utilizara Junit ,se crea un paquete src/test/java con la clase ApplicationTest.java

Se modifica la clase ApplicationTest.java para hacer pruebas insertando el objeto MockMvc, este objeto nos dará el método perform para ir probando los servicios y el resultado lo pintara en la consola

Se agrega @AutoConfigureMockMvc

@Autowired

MockMvc mock;

Y se añaden los métodos de prueba

@AutoConfigureMockMvc

@SpringBootTest

@TestMethodOrder(OrderAnnotation.class)

class ApplicationTests {

@Autowired

MockMvc mock;

//PRUEBA JUNIT ELIMINANDO UN CURSO

@Test

@Order(0)

void eliminarCurso()throws Exception{

mock.perform(delete("/curso/Phyton"));

}

//PRUEBA JUNIT OBTENIENDO TODOS LOS CURSOS

@Test

@Order(1)

void testCursos() throws Exception{

mock.perform(get("/cursos")).andDo(print());

}

@Test

@Order(2)

void testAlta() throws Exception{

mock.perform(post("/curso")

.contentType(MediaType.APPLICATION\_JSON)

.content("{\"nombre\":\"Angular 10\",\"duracion\":40,\"horario\":\"tarde\"}")

).andDo(print());

}

@Test

@Order(3)

void testActualizacion() throws Exception{

mock.perform(put("/curso")

.contentType(MediaType.APPLICATION\_JSON)

.content("{\"nombre\":\"Angular 10\",\"duracion\":80,\"horario\":\"mañana\"}")

).andDo(print());

}

}

Para ejecutar las pruebas, clic derecho sobre ApplicationTest.java

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Al ejecutarlo realiza las operaciones y se va viendo en consola

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En el apartado Body es donde nos muestra toda la informacion que trajo

**17. Utilización de la herramienta Postman**

También podemos utilizar Postman

Captura de pantalla de computadora

Descripción generada automáticamente