TEST UNITARIOS SIN MOCKS

1- Agregamos dependencias

2-En test/java/com.glubits.employees crear las siguientes carpetas:
integration
unit
utils

3- La carpeta unit contendra las carpetas:
repository
service

4- La carpeta utils contendra:
DTOSFACTORYS
FACTORYS

5-Ir a classRepository.java hacer : alt insert/test para que genere los test de los metodos (colocarlo en la carpeta unit/repository)
Instanciar la clase que voy a usar ejemplo: DepartmentRepository departmentRepository = new DepartmentRepository(); (no va el autowired porque no hay contexto de spring)
EJEMPLOS:

```
class EmployeeRepositoryTest {
EmployeeRepository employeeRepository;

 @Test
```

```
void save(){
    // arrange
    Department entity = ;
    Integer expected = ;

    // act
    var result = departmentRepository.save(entity);

    // assert
    Assertions.assertEquals(expected,result);
}
```

```
   @Test
void save(){
// arrange
Department entity = DepartmentFactory.getDepartament();
Integer expected = 3;// debe coincidir con la cantidad de datos iniciales más 1
 // act
    var result = departmentRepository.save(entity);

    // assert
    Assertions.assertEquals(expected,result);
}
```

```
@Test
void delete() {
// arrange
Integer id = 2;
Integer expected = 2;

// act
    var result = departmentRepository.delete(id);

    // assert
    Assertions.assertEquals(expected, result);
}

@Test
void deleteWithNotExistentId() {
    // arrange
    Integer id = 2;

    // act & assert
    Assertions.assertThrows(NotFoundException.class,
```

```
            () -> departmentRepository.delete(id));
}
```

```
   @Test
void findById() {
// arrange
Integer id = 1;
Optional<Employee> expected = Optional.of(EmployeeFactory.getMarco());

 // act
    var result = employeeRepository.findById(id);

    // assert
    Assertions.assertEquals(expected, result);
}


@Test
void findByIdNotExistent() {
    // arrange
    Integer id = 3;
    Optional<Employee> expected = Optional.empty();

    // act
    var result = employeeRepository.findById(id);

    // assert
    Assertions.assertEquals(expected, result);
}
```

```
   @Test
void findByName() {
// arrange
String name = "Gabi";
List<Employee> expected = List.of(EmployeeFactory.getGabi());
 // act
    var result = employeeRepository.findByName(name);

    // assert
    Assertions.assertEquals(expected, result);
}
```

```
  @Test
void findByNameNotExistent() {
// arrange
String name = "Mati";
List<Employee> expected =List.of();

// act
    var result = employeeRepository.findByName(name);

    // assert
    Assertions.assertEquals(expected, result);
}


@Test
void listAll() {
    // arrange
    List<Employee> expected = List.of(//los que estan agregados en el json y deben estar e
n el orden del json
            EmployeeFactory.getGabi(),
            EmployeeFactory.getMarco(),
            EmployeeFactory.getJean());
    // act
    var result = employeeRepository.listAll();

    // assert
    Assertions.assertEquals(expected, result);
}
```

}

6- Setear entidad para eso crear json: En test/directory/resources/(ej: department.json):

```
[
{
"id": 0,
"name": "Gabi",
"age": 19,
"email": "gabimonzon@digitalhouse.com",
"address": "Av. Siempreviva 742"
},
{
"id": 1,
"name": "Marco",
"age": 24,
"email": "marcoavila@digitalhouse.com",
"address": "221B Baker Street"
```

```
},
{
"id": 2,
"name": "Jean",
"age": 25,
"email": "jeancardo@digitalhouse.com",
"address": "Andén 9 y 3/4"
}
]
```

7-Ir a la carpeta utils y cread Factory :

```
public class EmployeeFactory {

public static Employee getTincho(){
    return Employee.builder()
            .name("Tincho")
            .age(30)
            .address("Av. BuenaVista 222")
            .email("martinmarquez@digitalhouse.com")
            .build();
}

public static Employee getMarco(){
    return Employee.builder()
            .id(1)
            .name("Marco")
            .age(24)
            .email("marcoavila@digitalhouse.com")
            .address("221B Baker Street")
            .build();
}

public static Employee getGabi(){
    return Employee.builder()
            .id(0)
            .name("Gabi")
            .age(19)
            .email("gabimonzon@digitalhouse.com")
            .address("Av. Siempreviva 742")
            .build();
}

public static Employee getJean(){
    return Employee.builder()
            .id(2)
            .name("Jean")
            .age(25)
            .email("jeancardo@digitalhouse.com")
```

```
            .address("Andén 9 y 3/4")
            .build();
}
}
```