TEST UNITARIOS CON MOCKS

1- La carpeta unit contendra las carpetas:
Dentro de service vamos a crear la clases de los servicio que corresponden:alt insert/test para que genere los test de los metodos (colocarlo en la carpeta unit/service)

EJEMPLOS:

```
@ExtendWith(MockitoExtension.class) //SIEMPRE SE PONE
class EmployeeServiceTest {
  @Mock
  EmployeeRepository employeeRepository; //dependencia instanciada

 @InjectMocks
EmployeeService employeeService; // a quien le inyectamos los test con mocks
```

```
 @Test
void saveEmployee() {
// arrange
EmployeeDTO employeeDTO = EmployeeDTOFactory.getTinchoDTO(); //LO ESPERADO
CrudDTO expected = CrudDTOFactory.crudDTOCreationWithId1(); //LLAMAMOS AL CRUD QUE CREAMOS
Employee entity = EmployeeFactory.getTincho(); //MANDAR LO DE LA FACTORY, EN ESTE CASO LA
 ENTITY ES LO QUE DEBERIAMOS MANDAR AL SERVICIO EN LA CONVERSION

   // act
        // whens
    Mockito.when(employeeRepository.save(entity)).thenReturn(0); //CUANDO SE EJECUTE ESTO
 DEVOLVEME ESTE VALOR
    var result = employeeService.saveEmployee(employeeDTO);

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

@Test
void deleteEmployee() {
    // arrange
```

```
    Integer id = 1; // necesita un entero
    CrudDTO expected = CrudDTOFactory.crudDTODeletationWithId1(); // se debe crear el fact
ory. lo que  espera es.

    // act con mocks
    var result = employeeService.deleteEmployee(id); //Llamo al metodo y le paso el id que
le corresponde
    Mockito.when(employeeRepository.delete(1)).thenReturn(1);// mockeo la llamada al repos
itory(puede haber varios when), luego de esto que retorne el valor que corresponde.
    // debe llegar el valor que le corresponde al metodo.

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

@Test
void listAllEmployees() { // los empleados los pasos a un factoryDto porque en el metodo l
os pasa a una lista dto.
    // arrange
    List<EmployeeDTO> expected = List.of(EmployeeDTOFactory.getGabiDTO(),//Lista de los em
pleados e la factory, por orden que fueron creados.
            EmployeeDTOFactory.getMarcoDTO(),
            EmployeeDTOFactory.getJeanDTO());

    List<Employee> shouldReturn = List.of(EmployeeFactory.getGabi(),//Lista de la entidad.
La factory comun
            EmployeeFactory.getMarco(),
            EmployeeFactory.getJean());

    // act
     var result = employeeService.listAllEmployees();//Llamada
    Mockito.when(employeeRepository.listAll()).thenReturn(shouldReturn);// Llamada al repo
sitory en el metodo no tiene parametro por eso es listAll(). siempre que
//que se llame a esto retorname la lista de la entidad

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

```
@Test
void findOneEmployee() {
    // arrange
    Integer id = 1;//Buscamos el empleado 1
    EmployeeDTO expected = EmployeeDTOFactory.getMarcoDTO();//Lo que devuelve , se espera.
    Employee marco = EmployeeFactory.getMarco();//Devolver una entidad en este caso

    // act
```

```
    var result = employeeService.findOneEmployee(id);
    Mockito.when(employeeRepository.findById(id)).thenReturn(Optional.of(marco));
                                                    // aca devuelve a marco. esta en un optio
nal porque findById devuelve un optional. se puede ver en employeerepository
    // assert
    Assertions.assertEquals(expected, result);
}

@Test
void findOneEmployeeWithNotExistentId() {
    // arrange
    Integer id = 999999;// id que no existe

    // mockeos

    Mockito.when(employeeRepository.findById(id)).thenReturn(Optional.empty()); // es empt
y porque esta vacio, no existe.

    // act & assert
    Assertions.assertThrows(   // Este seria nuestro expected, es una excepcion
            NotFoundException.class, // ver la excepcion que da el servicio
            () -> employeeService.findOneEmployee(id) // el ejecutable
    );
}

@Test
void findByName() {
    // arrange
    String name = "Jean"; //como parametro
    List<EmployeeDTO> expected = List.of(EmployeeDTOFactory.getJeanDTO());// lo esperado ,
en la lista lo que tiene que encontrar en este caso es un empleado
    Employee jean = EmployeeFactory.getJean(); // Entidad

    // act
    Mockito.when(employeeRepository.findByName(name)).thenReturn(List.of(jean));// retorno
una entidad
    var result = employeeService.findByName(name);

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

@Test
void findByNameNotFound() {
    // arrange
    String name = "Amarancio"; // El parametro que busca
    List<EmployeeDTO> expected = List.of(); // Lo esperado una lista vacia en este caso.Po
rque no tiene una excepcion el EmployeeService

    // act
    Mockito.when(employeeRepository.findByName(name)).thenReturn(List.of());//Retorna una
 lista vacia
    var result = employeeService.findByName(name);
```

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

2- En la carpeta utils crear la clase CrudDTOFactory: es un mensaje y una accion. EJEMPLOS:

```
public class CrudDTOFactory {

public static CrudDTO crudDTOCreationWithId1(){
    return CrudDTO.builder()
            .message("Se creo el empleado con id: " + 0)
            .action(CrudEnum.CREATION)
            .build();
}

public static CrudDTO crudDTODeletationWithId1(){
    return CrudDTO.builder()
            .message("Se elimino el empleado con id " + 1)
            .action(CrudEnum.DELETATION)
            .build();
}

public static CrudDTO crudDTOCreationWithId2(){
    return CrudDTO.builder()
            .message("Se creo el departamento con id: " + 2)
            .action(CrudEnum.CREATION)
            .build();
}

public static CrudDTO crudDTODeletationWithId2(){
    return CrudDTO.builder()
            .message("Se elimino el departamento con id " + 2)
            .action(CrudEnum.DELETATION)
            .build();
}

public static CrudDTO crudDTOCreationWithId3(){
    return CrudDTO.builder()
            .message("Se creo el empleado con id: " + 3)
            .action(CrudEnum.CREATION)
            .build();
}

public static CrudDTO crudDTOReadingNotFoundWithId7(){
```

```
    return CrudDTO.builder()
            .message("No se pudo encontrar el empleado con id " + 7)
            .action(CrudEnum.READING)
            .build();
}
```

}

3- Ir a la carpeta utils y cread Factory :
EJEMPLO:

```
public class EmployeeDTOFactory {

public static EmployeeDTO getTinchoDTO(){ //SIN ID PORQUE NOS LO DEVUELVE EL METODO saveEm
ployee
    return EmployeeDTO.builder()
            .name("Tincho")
            .age(30)
            .address("Av. BuenaVista 222")
            .email("martinmarquez@digitalhouse.com")
            .build();
}

public static EmployeeDTO getMarcoDTO(){

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

public static EmployeeDTO getGabiDTO(){

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

public static EmployeeDTO getJeanDTO(){
    return EmployeeDTO.builder()
            .id(2)
```

```
            .name("Jean")
            .age(25)
            .email("jeancardo@digitalhouse.com")
            .address("Andén 9 y 3/4")
            .build();
}
}
```