Pruebas Unitarias - Task Tracker

Pruebas unitarias realizadas sobre el servicio TaskService de la aplicación **Task Tracker**. El objetivo es garantizar que la lógica de negocio de las operaciones CRUD sobre las tareas funciona correctamente, usando **JUnit 5** y **Mockito**.

Servicio bajo prueba

**- Clase:** TaskService  
**- Paquete:** com.codes.tasktracker.demo.service  
- **Dependencia principal:** TaskRepository

Funciones a la que le vamos a realizar la pruebas Unitarias

* createTask(String description)
* getTask(UUID id)
* markTaskCompleted(UUID id)
* listAllTasks()
* updateTask(UUID id, String description, Boolean completed)
* deleteTask(UUID id)

Caso de Prueba 1: createTask

@Transactional  
public Task createTask(String description) {  
 Task task = new Task(description);  
 return repository.save(task);  
}

Objetivo de la prueba

Verificar que al crear una tarea:

* Se guarda en el repositorio con la descripción proporcionada.
* La tarea se marca como **no completada (completed=false)** por defecto.
* El objeto devuelto por el servicio contiene esos mismos valores.

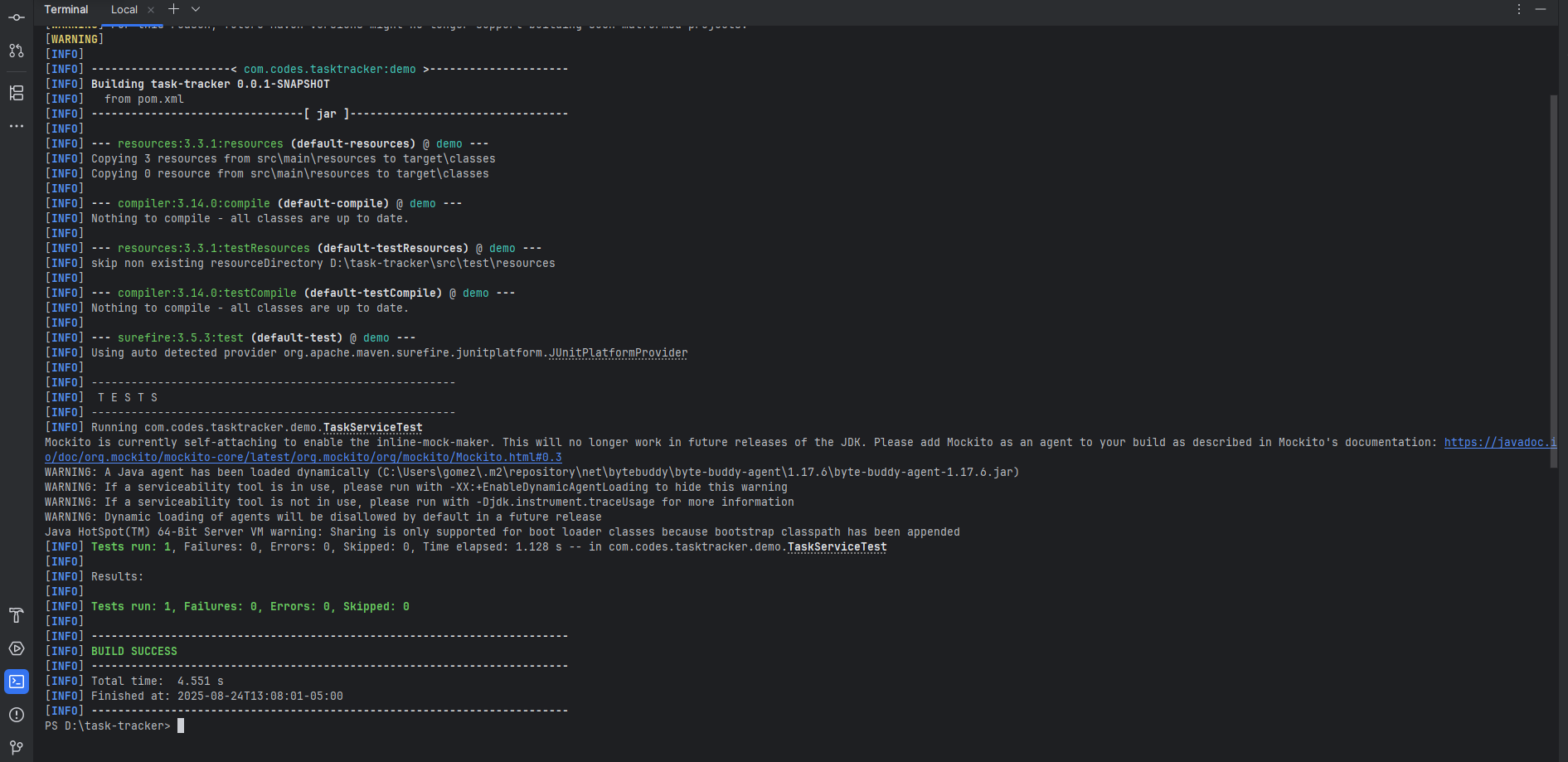
Implementación del test

@Test  
void createTask\_createsAndSavesWithDefaults() {  
 // Arrange: devolvemos el mismo objeto que recibe el repository  
 *when*(repository.save(*any*(Task.class))).thenAnswer(inv -> inv.getArgument(0));  
   
 Task result = service.createTask("Primera tarea");  
  
 // Assert: se guardó con la descripción y completed=false  
 *verify*(repository).save(taskCaptor.capture());  
 Task saved = taskCaptor.getValue();  
  
 *assertThat*(saved.getDescription()).isEqualTo("Primera tarea");  
 *assertThat*(saved.isCompleted()).isFalse();  
  
 // El resultado que devuelve el servicio coincide  
 *assertThat*(result.getDescription()).isEqualTo("Primera tarea");  
 *assertThat*(result.isCompleted()).isFalse();  
}

Resultado

Test ejecutado correctamente.  
La función createTask cumple con lo esperado: guarda la tarea con la descripción dada y completed=false por defecto.

Comando para correr el test : mvn -Dtest=TaskServiceTest test



Caso de Prueba 2: getTask cuando no existe la tarea

public Task getTask(UUID id) {  
 return repository.findById(id)  
 .orElseThrow(() -> new IllegalArgumentException("Tarea no encontrada: " + id));  
}

Objetivo de la prueba

Verificar que al intentar obtener una tarea que no existe:

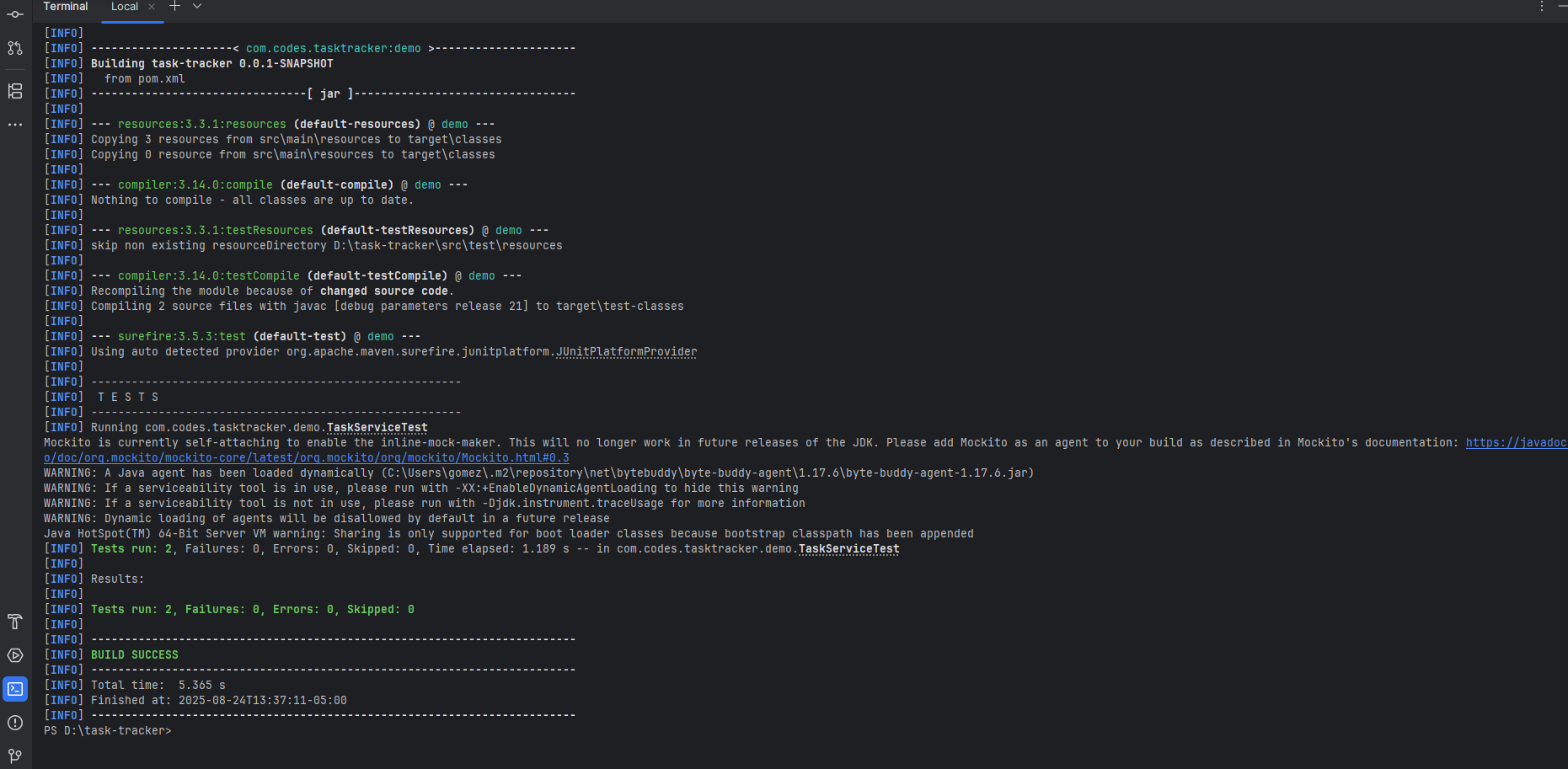
* El servicio lanza una excepción IllegalArgumentException.
* El mensaje de error contiene el texto "Tarea no encontrada".
* Se realiza la llamada al repositorio con el ID correcto.

Implementación del test

@Test  
void getTask\_throwsWhenNotFound() {  
 // Preparamos el repositorio para que devuelva vacío (simula que no existe la tarea)  
 UUID fakeId = UUID.*randomUUID*();  
 *when*(repository.findById(fakeId)).thenReturn(Optional.*empty*());  
   
 // Aquí esperamos que el servicio lance una IllegalArgumentException  
 Exception ex = *assertThrows*(IllegalArgumentException.class, () -> {  
 service.getTask(fakeId);  
 });  
  
 // Verificamos que el mensaje de la excepción contiene el texto esperado  
 *assertThat*(ex.getMessage()).contains("Tarea no encontrada");  
  
 // Verificamos que el repositorio fue llamado con el ID correcto  
 *verify*(repository).findById(fakeId);  
}

Resultado

Test ejecutado correctamente.  
La función getTask lanza la excepción esperada cuando el ID no existe en el repositorio.



Caso de Prueba 3: markTaskCompleted

// --- Mé*todo para ejemplo de marca completada ---*  
@Transactional  
public void markTaskCompleted(UUID id) {  
 Task task = getTask(id);  
 task.markCompleted();  
 repository.save(task);  
}

Objetivo de la prueba

Verificar que al marcar una tarea como completada:

* El campo completed cambia a true.
* El repositorio guarda la tarea modificada.
* Se invoca el método findById con el ID correcto.

@Test  
void markTaskCompleted\_marksAndSaves() {  
 //Creamos un UUID ficticio y una tarea inicial sin completar  
 UUID taskId = UUID.*randomUUID*();  
 Task existing = new Task("Tarea pendiente"); // por defecto completed=false  
  
 // Simulamos que el repositorio devuelve esta tarea cuando se busca por ID  
 *when*(repository.findById(taskId)).thenReturn(Optional.*of*(existing));  
 // Simulamos que el repositorio devuelve la tarea actualizada al guardar  
 *when*(repository.save(*any*(Task.class))).thenAnswer(inv -> inv.getArgument(0));  
  
 // Llamamos al mé*todo del servicio*  
service.markTaskCompleted(taskId);  
  
 // La tarea debe estar marcada como completada  
 *assertThat*(existing.isCompleted()).isTrue();  
  
 // Se debe haber guardado la tarea modificada en el repositorio  
 *verify*(repository).save(existing);  
}

Resultado

Test ejecutado correctamente.  
La función markTaskCompleted marca la tarea como completada y la persiste en el repositorio.

