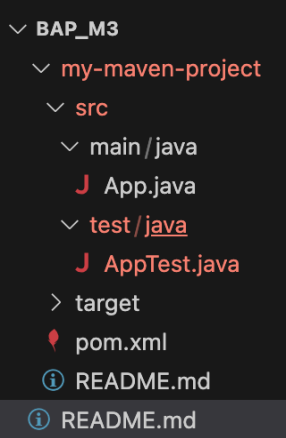
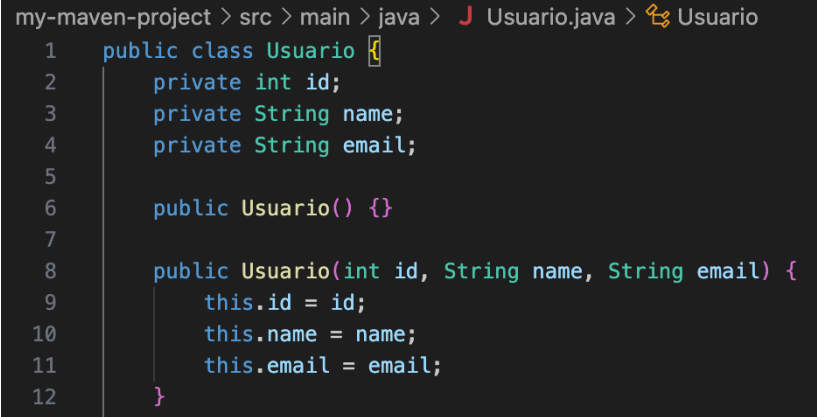
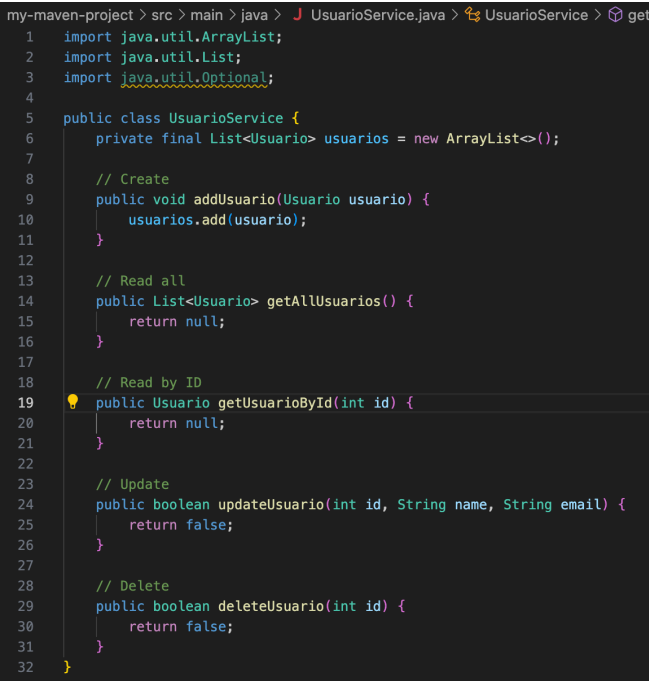



## Modulo 3: Evaluación - Proyecto: Módulo de Funcionalidades Básicas con TDD (CRUD)

Acciones ejecutadas	Detalle
Instalé en el PC	Visual Studio Code 1.101.1, Java 21, Java RTE (Temurin)
En VSCode instalé	PlugIn de Java, Maven, JUnit
<p>En VSCode cree un proyecto Java para Maven</p> 	<p>Cree una clase Usuario básica</p> 
<p>Cree la clase UsuarioService sin métodos desarrollados (return null y false)</p> 	<p>Cree la clase UsuarioServicetest para dar inicio al TDD</p> 

Agrego la dependencia para JUnit al POM.XML

```
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter</artifactId>
  <version>5.10.2</version>
  <scope>test</scope>
</dependency>
```

Ejecuto los tests con mvn test cumpliendo la etapa RED

```
[INFO]
[ERROR] Tests run: 7, Failures: 3, Errors: 2, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 1.605 s
[INFO] Finished at: 2025-07-05T16:40:14-04:00
[INFO]
```

Codifico los métodos de UsuarioService y ejecuto los tests cumpliendo la etapa GREEN

```
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.011 s
[INFO] Finished at: 2025-07-05T16:45:30-04:00
[INFO]
```

Para iniciar etapa REFACTOR que incluye persistencia se procederá con:

- Modificar el programa para que se conecte a SQLite (Ajuste al POM)
- Crear una base de datos SQLite local
- Crear en esta una Tabla de Users
- Modificar UserServiceTest para que inicialice esta Tabla en cada ejecución de los Tests ya existentes.
- Modificar los métodos de UserService para que usen la Base de Datos

Agrego la dependencia para SQLite a POM.XML

```
<dependency>
  <groupId>org.xerial</groupId>
  <artifactId>sqlite-jdbc</artifactId>
  <version>3.45.3.0</version>
</dependency>
```

Codifico los métodos de UsuarioService y ejecuto los tests cumpliendo la etapa REFACTOR exitosamente

```
UsuarioServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.748 s
[INFO] Finished at: 2025-07-05T20:38:11-04:00
[INFO]
```

Agrego codigo a App.java para demostrar que el CRUD funciona adecuadamente. Ejecuto con mvn exec:java -Dexec.mainClass=App;

```
**** 5 usuarios creados.
**** Lista de usuarios:
Usuario{id=1, name='Juan', email='juan@email.com'}
Usuario{id=2, name='Ana', email='Nuevo@email.Com'}
Usuario{id=3, name='Luis', email='luis@email.com'}
Usuario{id=4, name='Maria', email='maria@email.com'}
Usuario{id=5, name='Pedro', email='pedro@email.com'}
**** Usuario aleatorio:
Usuario{id=4, name='Maria', email='maria@email.com'}
**** Usuario actualizado: Usuario{id=4, name='Maria', email='Nuevo@email.Com'}
**** Usuario eliminado: Usuario{id=4, name='Maria', email='Nuevo@email.Com'}
**** Lista de usuarios después de eliminar:
Usuario{id=1, name='Juan', email='juan@email.com'}
Usuario{id=2, name='Ana', email='Nuevo@email.Com'}
Usuario{id=3, name='Luis', email='luis@email.com'}
Usuario{id=5, name='Pedro', email='pedro@email.com'}
```

Se procede a incluir Mock con Mockito  
Se incluye la dependencia en el POM.xml

```
<dependency>
  <groupId>org.mockito</groupId>
  <artifactId>mockito-core</artifactId>
  <version>5.2.0</version>
  <scope>test</scope>
</dependency>
```

Se define una clase ExampleTest haciendo uso de Mockito

```
public class ExampleTest {

    @Test
    public void testWithMock() {
        // Mock the UsuarioService class
        UsuarioService mockService = mock(classToMock:UsuarioService.class);

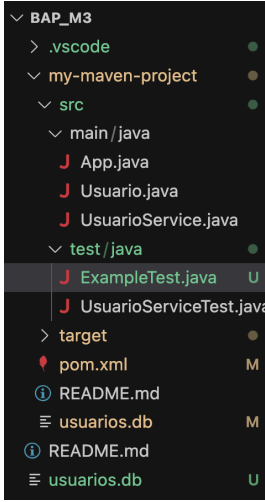
        // Define behavior for the mock
        when(mockService.getUsuarioById(id:1))
            .thenReturn(new Usuario(id:1, name:"Mock", email:"mock@email.com"));

        // Use the mock
        Usuario usuario = mockService.getUsuarioById(id:1);

        // Assert the mock behavior
        assertNotNull(usuario);
        assertEquals(expected:"Mock", usuario.getName());
        assertEquals(expected:"mock@email.com", usuario.getEmail());

        // Verify interaction
        verify(mockService).getUsuarioById(id:1);
    }
}
```

Se muestra como va la estructura del proyecto



Se inserta el PLUGIN para Jacoco

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.11</version>
  <executions>
    <execution>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>report</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```