Testing

In this app, we have several test types:

- **`@DataJpaTest`** "sliced" integration tests are used for repository testing.
- For validation services **`JUnit`** testing with **`Mockito`** library is used.
- For services, integration tests are used with **`@SpringBootTest`** and **`webEnvironment`** set to **`NONE`**. In that case Spring Boot creates an application context that does not contain servlets. There is no need for servlet layer when we test service layer method.
- To test controller layer the **`@WebMvcTest`** "sliced" tests are used. If we don't want to test other parts of the application, we mock those services with the **`@MockBean`** annotation.
- When we are writing integration tests that verify our application by accessing one of our HTTP endpoints we use configuration **`@SpringBootTest`** and **`webEnvironment`** set to **`RANDOM_PORT`**. Spring creates **`WebApplicationContext`** for our integration tests and starts the embedded servlet container on a random port. We can inject auto-configured HTTP (**`TestRestTemplate`**) clients that point to the started application.

The real test database "**test_intro_telecom**" is used for particular types of testing (for repository, service layer, the whole application). Test properties are defined in "**application-test.properties**" file.

"**create_test_db.sh**" – the script file that creates test database

"**test_intro_telecom_01_database.sql**" – database creation

"**test_intro_telecom_02_tables.sql**" – tables creation

"**test_intro_telecom_03_basic_data.sql**" – basic data for roles, package plans, add-on codes and SDR codes

"**test_intro_telecom_04_test_data.sql**" – for test data