D03 – Performance testing
Evaluation procedure

- The lecturers will check the following general sufficient conditions for failure, if applicable:
    - A legal interaction with your system results in an HTTP error.
    - A legal interaction with your system results in a panic.
    - Submitting a form with wrong data is not detected.
    - There are missing CRUD use cases.
    - An actor can list, edit, or delete data that belongs to another actor.
- Regarding management, the lecturers will check that:
    - You've followed the delivery instructions that are provided in document "On your deliverables".
    - Your project was properly managed. They'll pay special attention to checking that your tasks make sense, that they were early, and that you've allocated enough time to study the lessons, to work on the problems, the deliverable, to discuss it with your partner, and so on.
- Regarding your Eclipse/Maven projects, the lecturers will check that:
    - You've instantiated and customised the project template according to the guidelines that we provided to you. They'll pay special attention to checking that the databases and the URLs are in accordance with the name of the project.
- Regarding the models, the lecturers will check that:
    - You use your customer's vocabulary, which uses English terms.
    - The conceptual model represents the requirements faithfully.
    - The conceptual model doesn't have any void attribute.
    - The conceptual model's not been scaffolded.
    - The UML domain model doesn't have any artefacts that aren't supported by Java.
    - The UML domain model's not been scaffolded.
    - The Java domain model is clean and efficient.
    - The Java domain model includes every annotation that is required, including @NotNull and @Valid.
    - You use wrapper and primitive types correctly in your Java domain model.
- Regarding your persistence model, the lecturers will check that:
    - It actually represents your UML domain model faithfully.
    - Your "PopulateDatabase.xml" file specifies enough objects of each type and it accounts for enough variability, e.g., if entity "A" can be related to several entities of type "B", then they'll check that your "PopulateDatabase.xml" specifies an "A" entity that is related to as many "B" as possible.
    - Utility "PopulateDatabase.java" can be executed from within Eclipse and it persists the entities in your "PopulateDatabase.xml" file correctly.
    - Utility "QueryDatabase.java" can execute your JPQL queries and that they return the expected results, which you must properly document.
- Regarding your repositories and services, the lecturers will check that:
    - The queries in your repositories are simple and correctly implement the desired semantics. They'll review the functional requirements and will check that every such

requirement can be implemented with the queries that you provide in your repositories.
    - Your services are declared as transactional services, that you don't declare any static members or attributes other than the required autowired repositories and services, that no service manages a repository other than the one that it's intended to manage, and that business rules are implemented correctly. They'll review the functional requirements to check that your services provide the appropriate methods to implement them.

- Regarding your views, the lecturers will check that:
    - Your mock-ups follow the guidelines provided in this lesson and are appropriate to implement the corresponding functional requirements or use cases.
    - You've updated your Java domain model with appropriate "@DataTimeFormat" annotations, where necessary.
    - You've followed the guidelines that are provided in this lecture to implement views.
    - You've reused views appropriately when implementing similar requirements.
    - Your i18n&l10n bundles are correct.
    - Your Apache Tiles configuration files combine your views and your master page appropriately.

- Regarding your controllers, the lecturers will check that:
    - They rely on the appropriate services.
    - They implement well the listing and edition patterns that we've taught in the lectures.
    - The configuration files regarding security are properly configured.

- Regarding your system, the lecturers will:
    - Use credentials of the form *role-i*/*role-i* to log in to the system, where *role-i* denotes a role in your system. Typical credentials include *customer1*/*customer1*, *user2*/*user2*, *auditor3*/*auditor3*, and the like. The only exception is the default administrator's account, which must have the following credentials: *admin*
    - Check that every view works in both Spanish and English.
    - Check that that no form allows to enter invalid data, e.g., leaving blank fields that correspond to non-optional attributes, entering invalid dates or prices, or entering invalid enumerated values.
    - Check that when a form with validation errors is submitted, the information the user's entered remains in the form (that is, no field is cleared on entering invalid data).
    - Check that forms work well after entering invalid data, that is, if the incorrect fields are corrected, then the "save", the "delete", and the "cancel" button will work as expected.
    - Check that the "cancel" buttons work well in every edition form.
    - Check that pagination links work correctly. It's very important that your "PopulateDatabase.xml" file provides enough objects to overflow the listings. We assume that you'll set the default pagination to five records per page.
    - Check that it's not possible to hack the URLs of your application. They'll try to edit data that belongs to users other than the principal. They'll also try to have access to URLs that are not allowed to the principal.
    - Check that all of the information, functional, and non-functional requirements are implemented correctly.

- Regarding deployment, the lecturers will:

    - Check the report in which you describe how you've set up your pre-production configuration.
    - Check your script to create the database. Special attention will be paid to checking that it is executed within the context of a transaction, that the appropriate MySQL users are created and assigned the appropriate privileges on the database, and that the script does not introduce any sample or spurious data in the database, but the data required to implement a pre-defined *admin*/*admin* user account with administrative privileges and other pre-defined data that the system requires to run.
    - Check the script to delete the database. Special attention will be paid to checking that the MySQL users and their grants are removed completely.
    - Check your war artefact. Special attention will be paid to checking that it does not include your source Java code.
    - Execute the script to create your database in the pre-production environment.
    - Upload your war artefact to the Tomcat service in the pre-production configuration.
    - Check that the functional requirements described in the project statement work as expected and that there are no problems with your application when it's run on domain "www.acme.com".

- Regarding the loose ends, the lecturer will:
    - Check that your system complies with the following EU GDPR clauses: a) everything must be explained in plain terms; b) privacy must be enforced by design; c) there must be a mechanism to inform actors of security breaches; d) there must be a mechanism that an actor can use to access all of his or her data; e) there must be a mechanism to inform the actors of how the system processes their data; f) there must be a mechanism that an actor can use to remove all of his or her data from the system; g) there must be a mechanism that an actor can use to export his or her data. You are not requested to hire a data protection officer.
    - Check that your JSP views use appropriate custom tags.
    - Check that your entities include appropriate indices so as to boost your queries.
    - Check that your system cannot be GET, POST, or XSS hacked and that it does not suffer from SQL injections.

- Regarding functional testing, the lecturers will:
    - Check that you've followed the guidelines that we've provided regarding how to organise test cases, test classes, and test suites.
    - Run your test suite and will check that JUnit does not report any exceptions, that is, it must show a green bar.
    - Go through your test cases and will check that they are properly documented.
    - Check that your test cases are completely automatic, that is, that they do not output any information to the console, and that every check is performed by means of the appropriate assertion.
    - Check that their coverage is good enough both regarding statements and data.

- Regarding performance testing, the lecturers will:
    - Check that you have produced at least a performance test per use case.
    - Check that your scripts don't have any spurious entries.
    - Check that you've added timers to simulate sensible human delays when an interaction is started by a person, not by your browser.
    - Check that your report regarding the maximum workload that your system can handle does not show any HTTP errors.