D10 – Performance testing

Please, fill in this form and submit it with your deliverable; the filled form is expected to be renamed as "Statement.pdf". Your deliverable won't be evaluated unless you provide the identification data, check the responsibility statements, and submit it by the deadline.

Identification data

_____ _____

URL of your deliverable in ProjETSII | Level (C, B, A, or A+)

_____ _____

Place, date | Group number

_____

Students' names and signatures

Responsibility statements

[ ] I'm an author of this deliverable. I haven't cheated in any way.

[ ] I've collaborated with my partners on producing this deliverable; in other words, neither have I ridden their coattails nor gobbled them up.

[ ] I've learnt from working on this deliverable, so that I can pass my control checks.

[ ] I've organised this deliverable according to the guidelines that are available in document "On your deliverables.pdf", which is available at the USE's e-learning platform.

[ ] I understand that my deliverable will be considered failed if I fail to meet any of its requirements, if I fail to submit it by the deadline, or if I do not deliver a filled copy of this form.

[ ] I understand that failing this deliverable amounts to failing the subject.

[ ] I understand that I must have a contingency plan and that submitting my deliverable very close to the deadline is likely to result in disaster.

Requirements for levels C, B, and A

Item 1. Deliver a report with the tasks that you've accomplished to produce this deliverable. The report must list the name of every task, the moment when it started, the moment when it finished, and the number of hours spent on it. It must also report on the total time spent in the project and the total cost.

Item 2. Deliver a conceptual model and a UML domain model regarding your project. Deliver an Eclipse/Maven project that fulfils the requirements in the Acme Barter project statement. The project must provide a functional testing suite that fulfils the following requirements:

a. You must select a use case that involves a listing and an edition requirement; for that use case, you must implement at least 10 test cases that provide a good enough coverage of the statements and the parameter boundaries in your code.

b. For the remaining use cases, you must implement at least one positive test case and two negative test cases. Every test case must be properly documented. Please, copy the functional requirement(s) that they are intended to test as a comment at the beginning of your test cases; add a short remark regarding what each test case is intended to test.

Item 3. Deliver a performance testing suite that fulfils the following requirements:

a. There must be at least one jMeter test case per use case.

b. There must be a report in which you analyse what the maximum performance of your system is. The report must include screenshots where appropriate.

Item 4. Deliver a script to create the corresponding database in the pre-production environment and a war artefact that implements your project. The war artefact must be deployable and runnable on domain "www.acme.com".

Requirements for level A+

In the theory lecture, we've studied how to record a script that emulates a user interacting with our web information systems. Unfortunately, the data that we enter in the forms is recorded, too, which means that every time a script runs, the same data is used once and once again. jMeter allows to use CSV files to provide data to the forms. To earn your A+, you must deliver the following items:

Item 5. Write a report in which you explain how this jMeter mechanism works. The report's expected to be 1 000-word long; illustrations are strongly recommended.

Item 6. Deliver at least a test case in which you use this mechanism.

Evaluation procedure

- Regarding the deliverable, the lecturers will:
  - Check that you've followed the delivery instructions that are provided in document "On your deliverables".
- Regarding management, the lecturers will:
  - Check your project in ProjETSII. They'll check that you have a project, and that you've created and reported on the appropriate tasks.
  - Pay special attention to checking that the tasks were created at reasonable moments and that the reports happened at reasonable moments.

- Regarding documentation, the lecturers will:
    - Check that you've produced a document with an estimate of the total number of hours you've spent in this project and the total cost expected.
    - Check that you've produced a good conceptual model.
    - Check that you've produced a good UML domain model.
    - Check that you've produced comments in your code where appropriate, e.g., to document complex queries.
- Regarding your Eclipse/Maven project, the lecturers will:
    - Check that both your project and your database are properly named, using hyphens where necessary.
    - Check that you've instantiated and customised the project template according to the guidelines that we provided to you. The lecturers will pay special attention to checking that the databases and the URL's are in accordance with the name of the project.
    - Check that they can start your project up using the instructions that are provided in document "On your deliverables".
- Regarding your models, the lecturers will:
    - Check that you use your customer's vocabulary, which is in English, not Spanish.
    - Check that the conceptual model represents the requirements faithfully.
    - Check that the conceptual model doesn't have any void attributes.
    - Check that the conceptual model's not been scaffolded.
    - Check that the UML domain model doesn't have any artefacts that aren't supported by Java.
    - Check that the UML domain model doesn't have any relationships that are inefficient to implement.
    - Check that the UML domain model's not been scaffolded.
    - Check that the Java domain model is clean and efficient.
    - Check that the Java domain model includes every annotation that is required to represent implicit constraints in the UML domain model.
    - Check that you use wrapper and primitive types correctly and you use @Valid and @NotNull annotations properly in your Java domain model.
    - Check that your Java domain model is updated with the appropriate "@DataTimeFormat" annotations, where necessary.
- Regarding the population of the database, the lecturers will:
    - Check that your persistence model represents your UML domain model faithfully.
    - Check that your "PopulateDatabase.xml" file specifies enough objects of each type, and that it provides enough variability, e.g., if entity A can be related to several entities of type B (0..1, 0..*, or 1..*), then they'll check that your "PopulateDatabase.xml" specifies an A entity that is related to zero B entities, an A entity that is related to one B entity, and so on.
    - Check that your JPQL statements work well using utility "QueryDatabase.java".
- Regarding your architectural components, the lecturers will:
    - Check that the queries in your repositories are simple and correctly implement the desired semantics. The lecturers will review the functional requirements and will check that every such requirement can be implemented with the queries that you provide in your repositories.
    - Check that your services are declared transactional, you don't declare any static members or attributes other than the required autowired repositories and services, no service manages a repository other than the one that it's intended to manage, and business rules are implemented correctly. The lecturers will review the functional requirements to check that your services provide the appropriate services to implement them.

    - Check that your views are correct. In particular, they'll check that you've followed the guidelines that they have provided to implement views and that you've reused views appropriately when implementing similar requirements.
    - Check that that your i18n&l10n bundles are correct.
    - Check your Apache Tiles configuration files to make sure you've combined your views and the master page appropriately.
    - Check your controllers to make sure that they rely on the appropriate services and that they implement well the listing and edition patterns that we've taught in our lectures.
    - Check that the configuration files regarding security are correct.
- Regarding your deployment artefacts, the lecturers will:
    - 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 data in the database, but the data required to implement a predefined admin/admin user account with administrative privileges and other essential configuration data, if any.
    - Check your 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 their pre-production environment.
    - Upload your war artefact to the Tomcat service in their 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 laws, the lecturers will:
    - Check that your project complies with the minimum requirements that we've presented in the lecture notes about the following laws: LOPD, LSSI, and Transpositions, except for keeping your communications secure if you opt not to earn an A+.
- Regarding JSP views, the lecturers will:
    - Check that you're using customs tags to make your views more compact and less error prone.
- Regarding query efficiency, the lecturers will:
    - Go through your repositories and check that you've defined appropriate indices for your domain entities.
- Regarding hacking, the lecturers will:
    - Go through your services to check that you check the principal in order to prevent GET hacking.
    - Analyse your views and form objects, as well as your controllers and services to check that you're preventing POST hacking.
    - Check that your forms do not suffer from SQL injection or cross scripting.
- 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.

- 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.
    - Check that your report regarding the maximum workload that your system can handle does not show any HTTP errors.
- Regarding the application, itself, the lecturers will:
    - Check that all of the information, functional, and non-functional requirements are implemented correctly.
    - Pay special attention to checking that registration forms require the user to enter the password twice, so as to minimise the chances that the user enters an invalid one that he or she cannot recover.
    - Check that every form 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 properties, entering invalid dates or prices.
    - Check that if a form has validation errors and you hit the "save" button, the information the user's entered remains in the form (that is, no field's 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, that is, they get the user back to the corresponding listing.
    - Check that pagination links work correctly in tables.
    - 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.
    - Check that the application works well when concurrency errors happen.