D11 – Acceptance testing Evaluation procedure

- 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 not created in a batch, and that you've followed your lecturers' guidelines.
- Regarding laws, the lecturers will check that:
  - Your project's in accordance with the Spanish LOPD law. The only exception can be regarding confidential and secure communications unless you wish to earn an A+.
  - Your project's in accordance with the Spanish LSSI law. The only exception is regarding informing the Chamber of Commerce about your internet domain.
  - Your project's in accordance with the Spanish Transpositions law.
- Regarding documentation, the lecturers will check that:
  - You've documented your project costs by means of a budget that makes sense.
  - Your code includes remarks where appropriate, and that these remarks are not trivial and really make sense.
- 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 URL's are in accordance with the name of the project.
  - You've customised the project template to adapt it to the Spanish regulations.
- Regarding your domain models, the lecturers will check that:
  - You use your customer's vocabulary.
  - 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 doesn't have any relationships that are inefficient to implement.
  - 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 to represent implicit constraints in the UML domain model.
  - You use wrapper and primitive types correctly in your Java domain model.
  - You use @Valid and @NotNull annotations properly.
- 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 that it accounts for 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" speci-

  - fies an A entity that is related to zero B entities, an A entity that is related to one B entity, and so on.
  - Utility “PopulateDatabase.java” can be executed from within Eclipse and 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 they 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 services to implement them.
  - You have written at least a check per service and method in that service, and that it makes sense. They expect that you write both positive and negative checks.
  - You check the principal in order to prevent GET or POST hacking.
  - You have added the appropriate methods to reconstruct your domain objects from form objects or pruned domain objects and that you use validators properly.
  - You’ve defined appropriate indices for your domain entities according to the queries in your repositories.

- Regarding your views, the lecturers will check that:
  - 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 combines your views and the master page appropriately.
  - You’ve made your best attempt to prevent POST hacking using appropriate form objects and pruned domain objects.
  - You’re using customs tags to make your views more compact and less error prone.
  - Your forms cannot suffer from cross scripting.

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

- Regarding your system, the lecturers will:
  - Use the credentials of the form “authority1/authority1”, “authority2/authority2”, “authority3/authority3”, and so on to log in to your system, where “authority” refers to any of the authorities in your system. The only exceptions are credentials “admin/admin”, which will be used to have access to your system as an administrator.
  - 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.

    - 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 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 not any problems with your application when it's run on domain "www.acme.com".
- 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 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.

- Regarding acceptance testing, the lecturers will:
    - Check that you have followed the guidelines provided by the lecturers to design appropriate checks for every use case.
    - Check that your intentional bugs make sense.