DS – September summon deliverable 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 in ProjETSII. They'll pay special attention to checking that your tasks make sense, that they were not created in a batch before the deadline, and that you've allocated 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 attributes.
    - 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 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 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.

- 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 combine your views and the master page appropriately.
- Regarding your controllers, the lecturers will check that:
    - 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 the lectures.
    - The configuration files regarding security are properly configured.
- Regarding your deployment, the lecturers will check that:
    - You have created the database artefact according to the guideline provided.
    - You have created the application artefact according to the guideline provided.
    - The artefacts that you provide can be deployed without any problems to the pre-production configuration that was provided during the first lesson.
    - The project was deployed to Clever Cloud and it starts up without any problems.
- Regarding your system, the lecturers will:
    - Use the following credentials to log in to the system: admin/admin, customer1/ customer1, customer2/customer2, handyworker1/ handyworker1, handyworker2/ handyworker2, referee1/referee1, referee2/referee2, sponsor1/sponsor1, and sponsor2/ sponsor2.
    - 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.