Juan Nicolás Camelo Garzón Cod: 257914

Fabián Darío Moreno Cod: 257424

Julián Felipe Cárdenas Cod: 257995

Giovanni Alexander León Cod: 258024

Jorge Andrés Serrano Jaimes Cod: 257452

Grails Practice:

1. **What is the H2 database console? What risk does it represents? What would you as a developer do to prevent those risks?:**

   H2 is a relational database management system written in Java. It has a console application that lets the user access to the database using a browser interface.

   Since it's an open source project, it would probably has bugs that have not yet been found. Also there are some features that are known to be dangerous. These are:

   - Disabling the transaction log or FileDescriptor.sync() using LOG=0 or LOG=1.
   - Using the transaction isolation level READ_UNCOMMITED (LOCK_MODE 0) while at the same time using multiple connections.
   - Disabling database file protection using (setting FILE_LOCK to NO in the database URL).
   - Disabling referential integrity using SET REFERENTIAL_INTEGRITY_FALSE.

   Also, running out of memory should be avoided, because in some older versions OutOfMemory errors while using the database could corrupt it. Another known risk is that when a user tries to connect to the database,

his/her username and password are hashed using SHA-256 and this value is transmitted to the database. So if an attacker is able to listen to the (unencrypted) transmission between the client and the database, it could get that username and password.

Some of the measures we as developers could take are:

a. Secure the channel of transmission between the client and the database, using different security measures and practices in order to avoid a listener to take place in the transmission.

b. Try not to use or be very careful when dealing with features explained above, adding constraints and restrictions to the users that can interact with them.

c. Check the hardware requirements periodically or when the database growths in order to avoid out of memory errors, and also make backups of it regularly.

2. When do you set up the database and the tables to use on your application?

Normally, the database is implemented first with all its tables before starting to create the Class Model, so it could be used as a guide for this. But if we are using a framework like Grails, this step it's not necessary because the framework automatically provides it when the Class Model is implemented.

3. In Grails, how it is replaced the XML configuration in your application? Give two motives why Grails solution improves the previous one.

In Grails, there is no need of XML configuration. The framework itself provides the XML files needed, using a set of rules or conventions while inspecting the code of Grails-based applications.
For example, a class name which ends with Controller is considered a web controller.

This feature results in great benefits to the developers, improving the productivity and focusing their attention in more important tasks.

4. **What is a domain instance? When Grails does validate a domain instance?**

   A domain represents the Model in the MVC (Model View Controller) pattern used by grails and represents an entity mapped onto an underlying database table. In Grails, it also represents a class that is in the `grails-app/domain` directory.

   By default, a class name is mapped to the table name in lower case and separated by underscores instead of camel case (ej. A domain class named TripPlanner would map to a table named trip_planner.

   Grails provides a unified way to define validation “constrains” with a constrain mechanism. This constrains declare specify validation rules most commonly applied to domain classes.
   Also, we could use the validate method to validate a domain class instance.

   The validation method has two phases:

   - Data binding: This occurs when we bind a request parameters onto an instance.

   - Call validate or save method: This is when Grails will validate the bound values against the constrains we already defined.

5. **When using scaffolding, it is possible to modify views or logic of a given domain class?**
   If the answer is yes, explain how can you do that, but if the answer is no, give one reason why this is not allowed.

   Yes, we can add new actions to a scaffolded controller or override them. We only need to access to the Controller class like any class we have already created. This is possible because scaffolding auto-generates views and controller actions for CRUD operations that we would have to implement by ourselves anyway.

6. **What can you do in order to have a set of data in Grails database to persist along several executions of the project, while in development?**

   Since the data model in Grails is written to the database using GORM (Grails Object Relational Mapping), domain classes are recorded in the directory `grails-app/domain`.
   So in order to have persisted data in the database between executions we could do the following:

   - Make a file with the data we need and put it in `grails-app/config/BootStrap.groovy`, so the database gets load with this data every time we run the application.

   - Change the database configuration in the file `grails-app/conf/DataSource.groovy` by changing the line `dbCreate ="create-drop"` with `dbCreate="update"` so the data don't get deleted every time we run the application.

7. **What are the differences between using findAll(), getAll() and list() and when it's useful or not to use each them for finding all of domain class instances?**

   - findAll() finds all domain class instances matching a specified query. Ej: `Book.findAll("from Book as b where b.author=?", ['Dan Brown'])`

   - getAll() retrieves a list of instances of the domain class for the specified ids, ordered by the original ids list. Ej: `def booklist = Book.getAll(1, 2, 3)`

   - list() list instances of the domain class. Ej: `def results = Book.list()`

   findAll() could be useful when we need certain instances of the domain class that have a specific value for a certain attribute like an SQL query. getAll() could be useful when we need certain instances in certain order defined by their ids.

   Finally list() could be useful when we need to create a view that shows the data stored in the database or so.

8. **What is the difference between using `p.save()` and p.save(flush: `true`, failOnError: `true`)?**

   Using `p.save`, saves a new domain class instance or updates a persistent one in the database, cascading up to his child instances if's required.

   But using `p.save(flush: true, failOnError: true)`, the object is immediately persisted and updated to the `version` column for optimistic looking (feature that involves storing a version value in a special `version` column in the database that is incremented after each update). Also, the method will throw a `grails.validation.ValidationException` if validation fails.