1. **Explains what the repository pattern consists of and what alternatives exist for its implementation.**

A **Repository** mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects. The interface is independent of the fields of classes A Repository provides an abstraction so that data can be accessed as if it was an in-memory collection, all data operations are made with interface methods without interact directly with database. There are mainly 2 approaches:

Repository per entity create a repository for each business object you need to store into persistence layer, then we implement only the methods that we need in our application

Generic repository consist in one implementation for each specific persistence technology instead of one implementation per domain class.

1. **The basic path testing technique guarantees a degree of coverage of condition. Explain what this type of coverage consists of and illustrate it with an example**
2. **Explain what the Unit of Work (UoW) persistence pattern consists of. Indicates what benefits it provides and its correspondence with the Entity Framework**

Unit of Work (UoW) persistence pattern consists in a list of objects affected by a business transaction, it coordinates the writing out of changes and the resolution of concurrency problems. the correspondence with Entity Framework is DBContext class, it is a bridge between entity classes and the database.

1. **Briefly explain the differences between a form viewed modally**

**(ShowDialog method) or modeless (Show method). When is it convenient**

**employ each of them? You can give as an example the one used in the practices**

Modal: It must be closed to return to the main form. It is shown using the method ShowDialog(), they are used for standar dialog boxes .

Non Modal: several forms may be used simultaneously. Shown using the method Show(), they are used for message boxes.

1. **What is Entity Framework? What advantages and disadvantages does its use imply?**

Entity Framework is an Object/Relational Mapping (O/RM) framework. It keeps our database design separate from our domain class design.

**PRO:** easy way to create database from a domain class (Code-first), generate data accesses classes for existing database (database first), create database and classes from the DB model design. (model first)

**CONTRO:** Its logical schema is not able to understand business entities and relation among each other, it does not work if we change any schema of the database. We need to update the schema on the solution.

1. **Define the term "Test Case", what is its objective and cite what techniques to design test cases exist.**

The test case is «a set of inputs, execution conditions and expected results developed for a given goal», mainly 2 techniques to design test cases:

White Box Testing: uses the procedural control structure to derive the test cases

Black Box Testing: focused on the functional requirements of software.

1. **Is EntityFramework an example of a DAO data access pattern? Reasos briefly answer.**
2. **In a UML use case diagram it is possible to indicate a relationship of**

**specialization among actors. What does this type of relationship indicate?**

Yes, it’s indicate that an actor son can do the same case of use of his parent + some case of use specialized for him.

1. **What development approaches does the Entity Framework support? What inputs do you require and what outputs does each of them offer?**

Generate data accesses classes for existing database (database first), INPUT tables OUTPUT domain classes

Create database from a domain class (Code-first), INPUT domain classes OUTPUT tables

Create database and classes from the DB model design (model first), INPUT DB model OUTPUT tables