SWE 203 - WEB PROGRAMMING

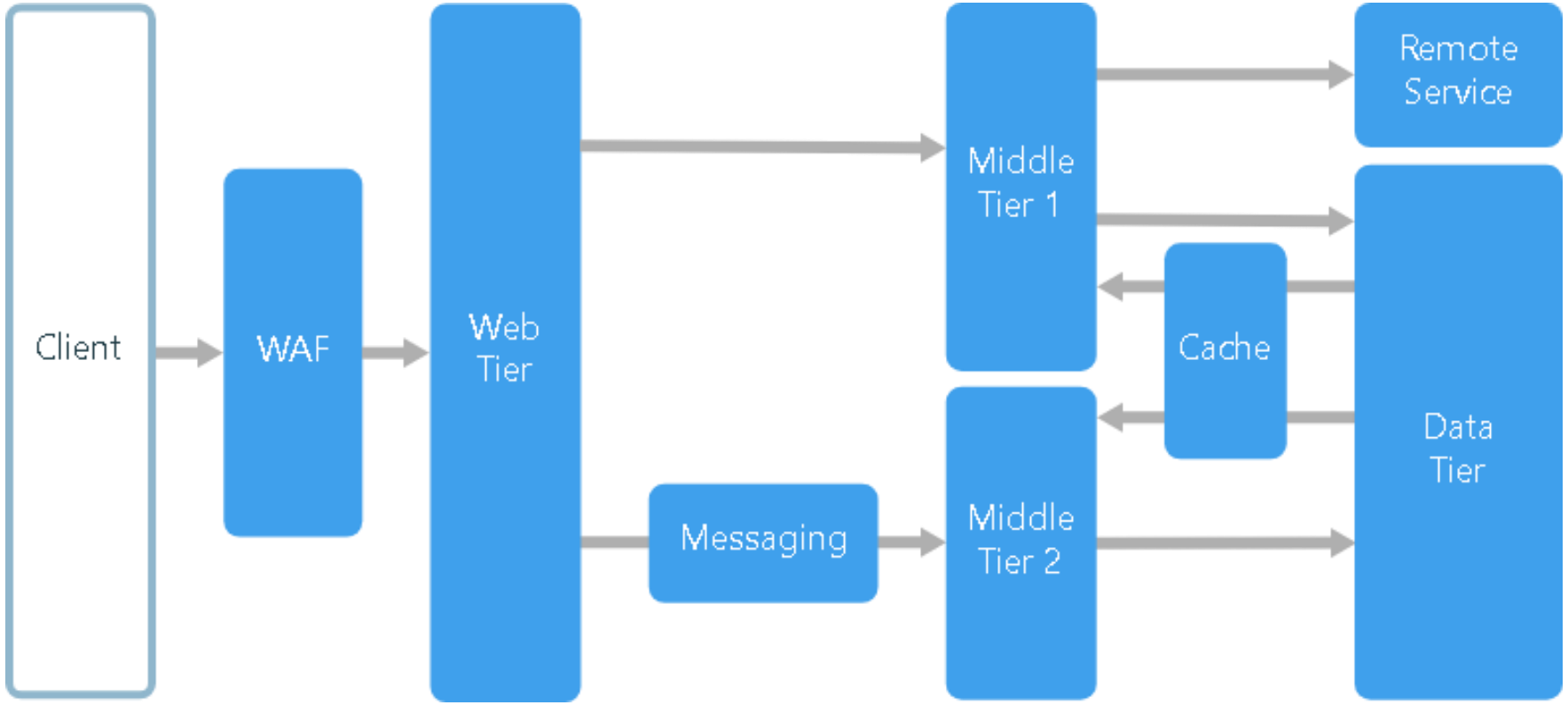
DR. DENIZ BALTA

ARCHITECTURES
WEEK 2

N-TIER ARCHITECTURE

- N-tier architecture is also called multi-tier architecture because the software is engineered to have the processing, data management, and presentation functions physically and logically separated.
- That means that these different functions are hosted on several machines or clusters, ensuring that services are provided without resources being shared and, as such, these services are delivered at top capacity.
- The "N" in the name n-tier architecture refers to any number from 1.

N-TIER ARCHITECTURE

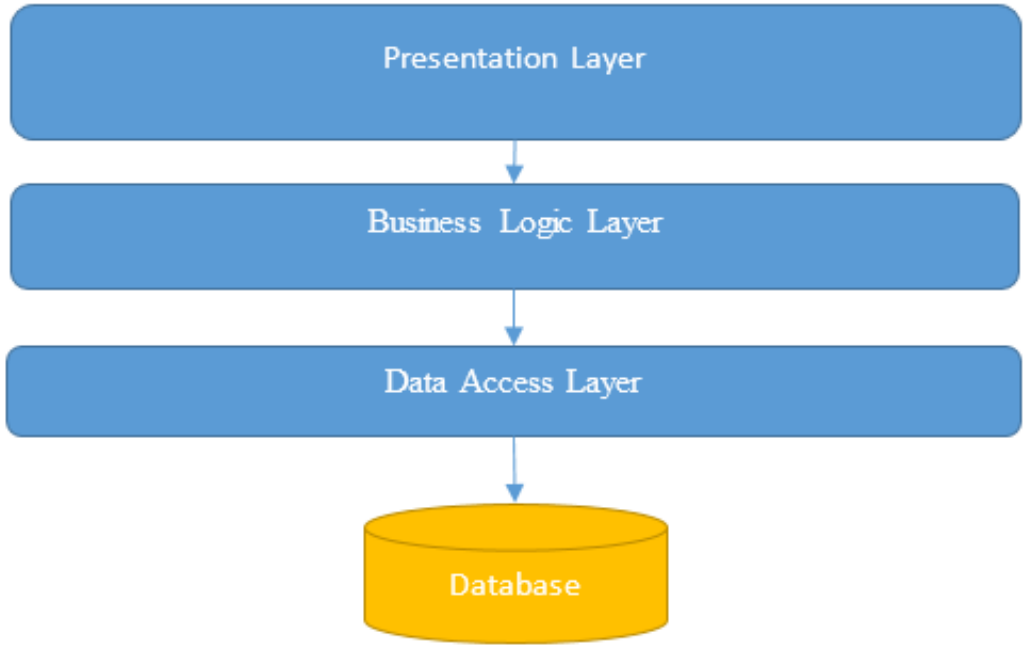


N-TIER ARCHITECTURE

N-tier architecture would involve dividing an application into three different tiers.

These would be the

- logic tier,
- the presentation tier, and
- the data tier.



WHAT ARE THE BENEFITS OF N-TIER ARCHITECTURE?

- **Secure:** You can secure each of the three tiers separately using different methods.
- **Easy to manage:** You can manage each tier separately, adding or modifying each tier without affecting the other tiers.
- **Scalable:** If you need to add more resources, you can do it per tier, without affecting the other tiers.
- **Flexible:** Apart from isolated scalability, you can also expand each tier in any manner that your requirements dictate.

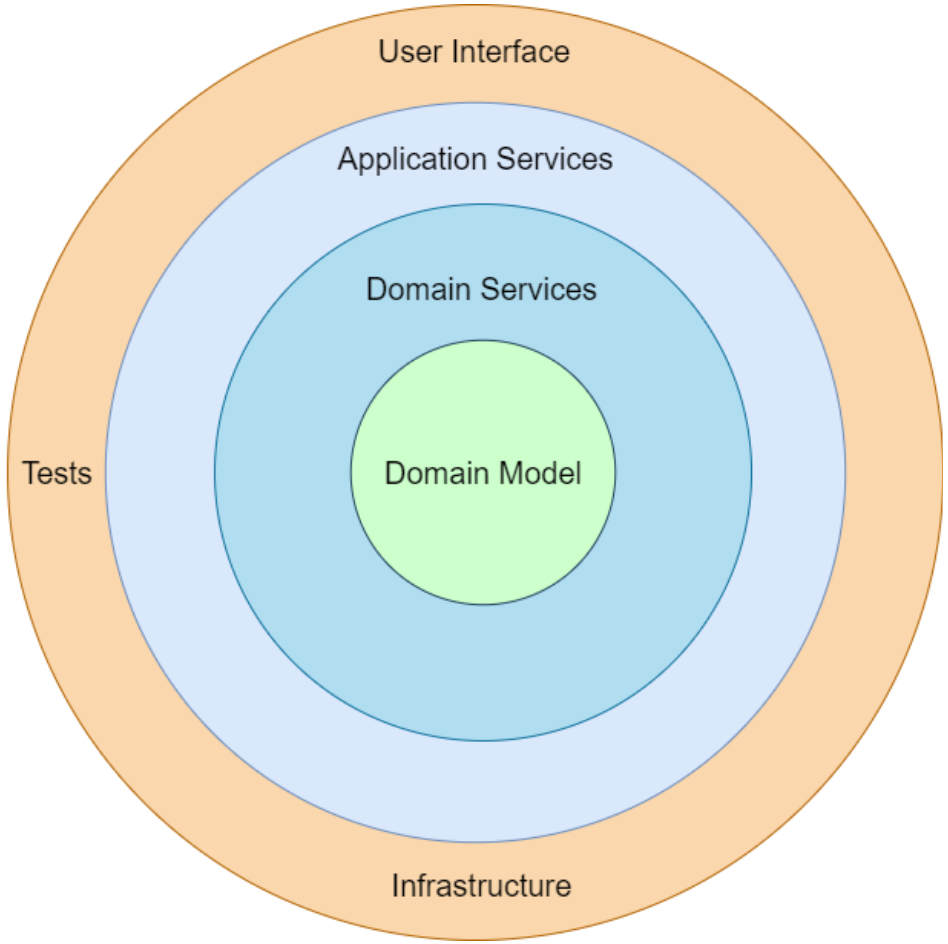
N-TIER ARCHITECTURE

- **The application logic tier.** The application logic tier is where all the "thinking" happens, and it knows what is allowed by your application and what is possible, and it makes other decisions. This logic tier is also the one that writes and reads data into the data tier.
- **The data tier.** The data tier is where all the data used in your application are stored. You can securely store data on this tier, do transaction, and even search through volumes and volumes of data in a matter of seconds.
- **The presentation tier.** The presentation tier is the user interface. This is what the software user sees and interacts with. This is where they enter the needed information. This tier also acts as a go-between for the data tier and the user, passing on the user's different actions to the logic tier.

DISADVANTAGES OF N-TIER ARCHITECTURE

- N-Tier layered architecture consists of layers that are tightly connected to each other.
- This can cause problems in the future, especially in large-scale projects.
- Data Access is usually the Core of the Entire Application.
- Any minor change in the Business Logic layer or the Presentation layer can be dangerous to the integrity of the entire application.

ONION ARCHITECTURE



WHAT IS ONION ARCHITECTURE?

- *Onion Architecture* was introduced by Jeffrey Palermo to provide a better way to build applications in perspective of better testability, maintainability, and dependability.
- Onion Architecture addresses the challenges faced with 3-tier and n-tier architectures, and to provide a solution for common problems.
- Onion architecture layers interact to each other by using the Interfaces.
- C# programmers are drawn to Onion Architecture due to the dependency flows.

ONION ARCHITECTURE PRINCIPLES

- Onion Architecture is based on the *inversion of control* principle.
- Onion Architecture is comprised of multiple concentric layers interfacing each other towards the core that represents the domain.
- The architecture does not depend on the data layer as in classic multi-tier architectures, but on the actual domain models.

WHAT ARE SOME PROBLEMS WITH ONION ARCHITECTURE?

- As per traditional architecture, the UI layer interacts to business logic, and business logic talks to the data layer, and all the layers are mixed up and depend heavily on each other.
- In 3-tier and n-tier architectures, none of the layers are independent; this fact raises a separation of concerns.
- Such systems are very hard to understand and maintain. The drawback of this traditional architecture is unnecessary coupling.
- Onion Architecture solved these problem by defining layers from the core to the Infrastructure.

WHAT ARE SOME PROBLEMS WITH ONION ARCHITECTURE?

- It applies the fundamental rule by moving all coupling towards the center.
- This architecture is undoubtedly biased toward object-oriented programming, and it puts objects before all others.
- At the center of Onion Architecture is the domain model, which represents the business and behavior objects.
- Around the domain layer are other layers, with more behaviors.

WHAT ARE THE LAYERS OF THE ONION ARCHITECTURE?

1. Domain Layer
2. Repository Layer
3. Services Layer
4. UI Layer

ONION ARCHITECTURE – DOMAIN LAYER

- At the center part of the Onion Architecture, the domain layer exists; this layer represents the business and behavior objects.
- The idea is to have all of your domain objects at this core.
- It holds all application domain objects.
- Besides the domain objects, you also could have domain interfaces.
- These domain entities don't have any dependencies.
- Domain objects are also flat as they should be, without any heavy code or dependencies.

ONION ARCHITECTURE –REPOSITORY LAYER

- This layer creates an abstraction between the domain entities and business logic of an application.
- In this layer, we typically add interfaces that provide object saving and retrieving behavior typically by involving a database.
- This layer consists of the data access pattern, which is a more loosely coupled approach to data access.
- We also create a generic repository, and add queries to retrieve data from the source, map the data from data source to a business entity, and persist changes in the business entity to the data source.

ONION ARCHITECTURE – SERVICES LAYER

- The Service layer holds ***interfaces*** with common operations, such as Add, Save, Edit, and Delete.
- Also, this layer is used to communicate between the UI layer and repository layer.
- The Service layer also could hold business logic for an entity.
- In this layer, service interfaces are kept separate from its implementation, keeping loose coupling and separation of concerns in mind.

ONION ARCHITECTURE – UI LAYER

- It's the outer-most layer, and keeps peripheral concerns like UI and tests.
- For a Web application, it represents the Web API or Unit Test project.
- This layer has an implementation of the dependency injection principle so that the application builds a loosely coupled structure and can communicate to the internal layer via interfaces.

BENEFITS AND DRAWBACKS OF ONION ARCHITECTURE

- Onion Architecture layers are connected through interfaces. Implantations are provided during run time.
- Application architecture is built on top of a domain model.
- All external dependency, like database access and service calls, are represented in external layers.
- No dependencies of the Internal layer with external layers.
- Couplings are towards the center.
- Flexible and sustainable and portable architecture.
- No need to create common and shared projects.
- Can be quickly tested because the application core does not depend on anything.

DRAWBACKS OF ONION ARCH.

A few drawbacks of Onion Architecture as follows:

- Not easy to understand for beginners, learning curve involved. Architects mostly mess up splitting responsibilities between layers.
- Heavily used interfaces