Capgemini

29th Aug 2023

LAYERED Vs HEXAGONAL ARCHITECTURE

Expert Connect Session # 1

Speakers

Sneha Sarmokadam

Architect
Europe C&CA

Rajesh Agarwal

Sr. Architect
Europe C&CA

AGENDA

- Updates from India Java Community
- Overview of Software Architecture
  - What is Architecture?
  - Architecture Principals
  - Architecture Patterns
- Overview of Layered Architecture
- Overview of Hexagonal Architecture
- Layered vs Hexagonal Architecture
- Summary
- Feedback & Quiz

GOAL

LEARN | UPSKILL | NETWORK

COMMUNITY UPDATES

Manjula Samuel

Level 2 Certified Sr. Architect
India Java Community Lead

INDIA JAVA COMMUNITY - UPDATES

	Description	Progress
Quiz Capsules	Quiz allowing individual participation on SpringBoot microservices topics	Season 1 of 8 Episodes has been launched Episode 3 of Season is in progress
Expert Speak Sessions	Provides an opportunity for associates to **connect with SME in a 1:1 session** and ask questions on a requested technical topic	Track has been launched today with first session today. Monthly Activity
Knowledge Sharing Session	Provides an opportunity for our Associates to know the **success stories** from people of successful engagements and learn the best practices from them.	Session being planned in Sep

Capgemini

LAYERED VS HEXAGONAL ARCHITECTURE

WHO AM I?

- 23+ years of IT Experience
- Roles
  - Lead Account Architect
  - Architect Community Lead (Germany BU)
  - India Java Community - Expert Connect Lead
- Certified Sr. Architect (L2)
- Expertise in
  - Problem Solving, System Design

Rajesh Agarwal

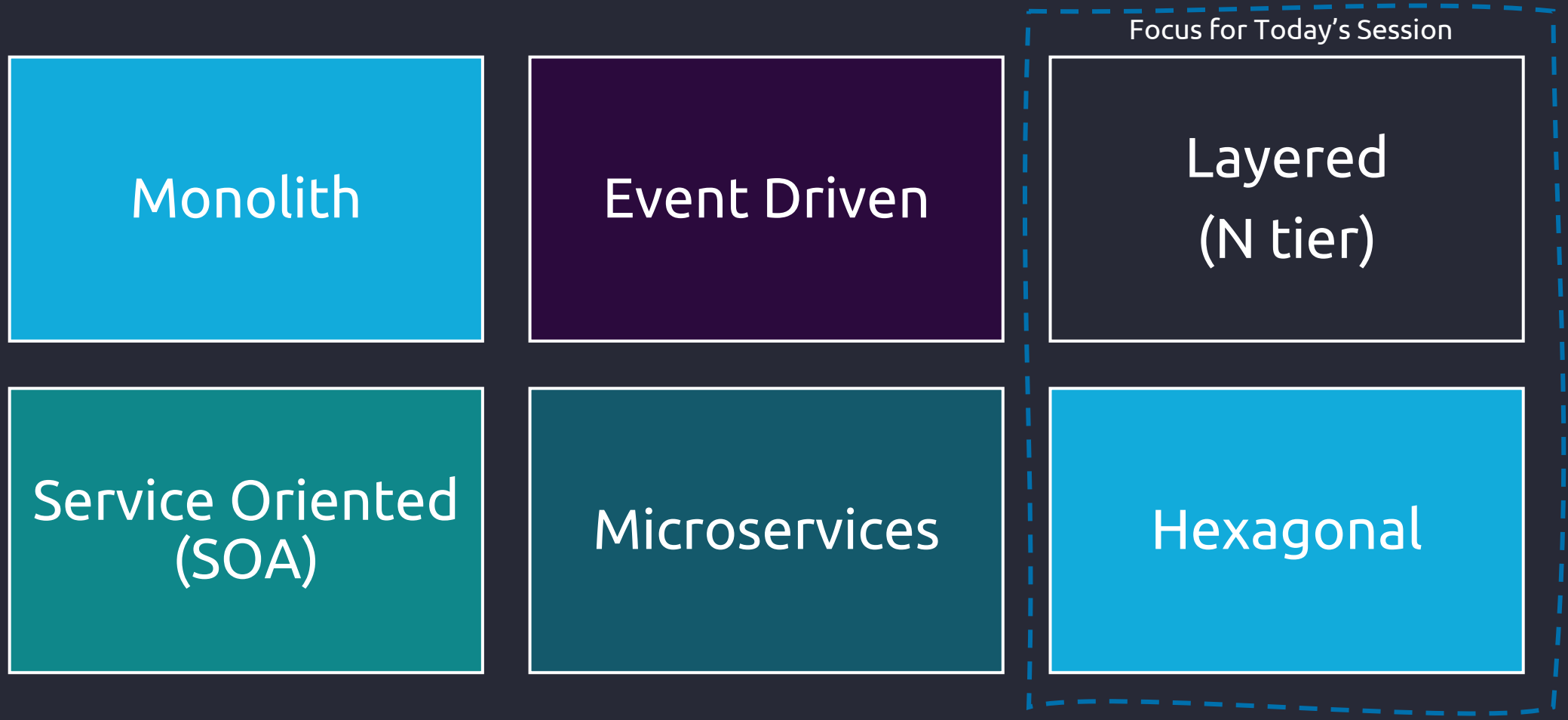
ARCHITECTURE

WHAT IS SOFTWARE ARCHITECTURE?

Software architecture is the structure of a software system, which consists of components/elements, their relationships, and the principles and guidelines governing their design and evolution.

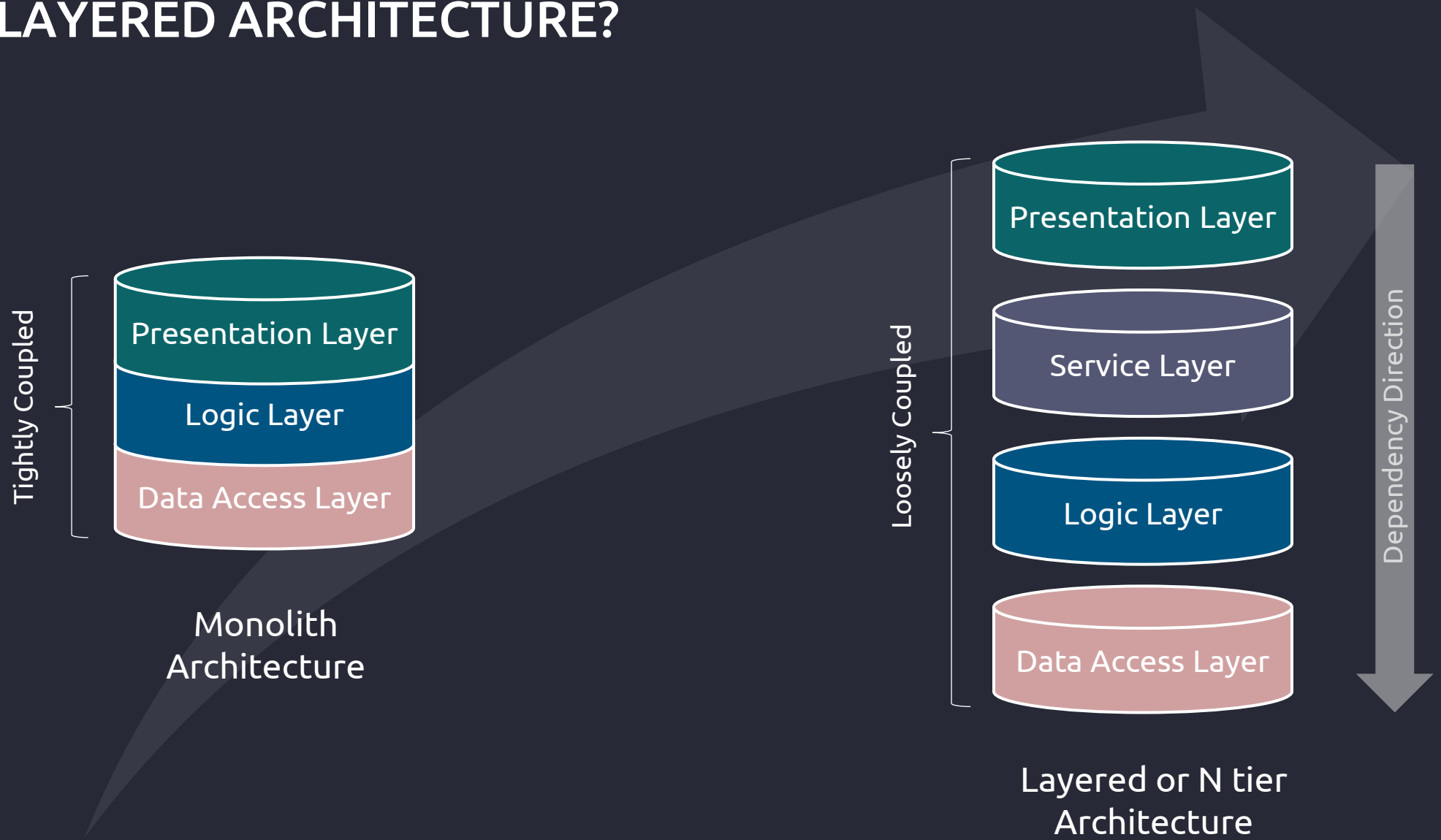
PRINCIPALS OF SOFTWARE ARCHITECTURE

ARCHITECTURE PATTERNS

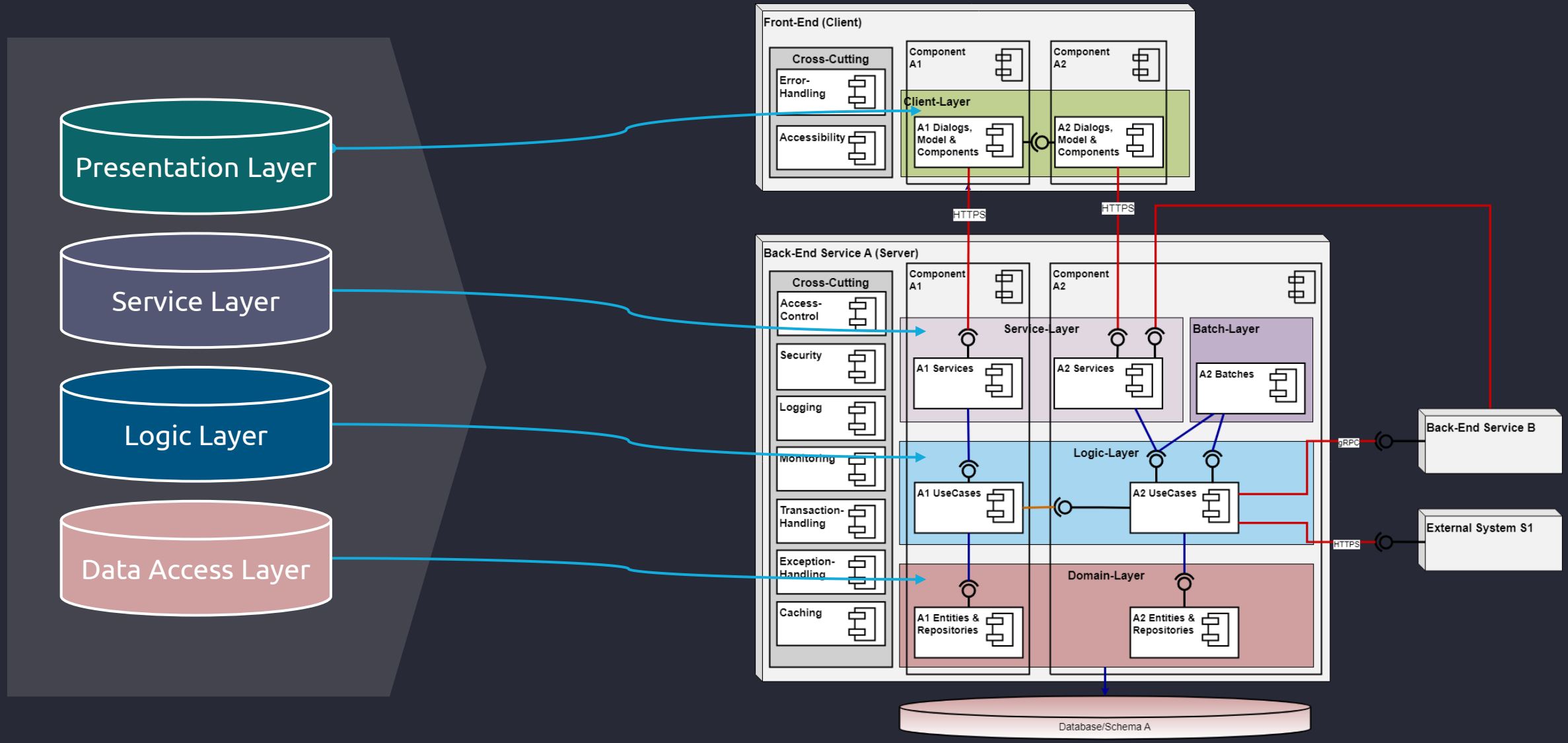


LAYERED ARCHITECTURE

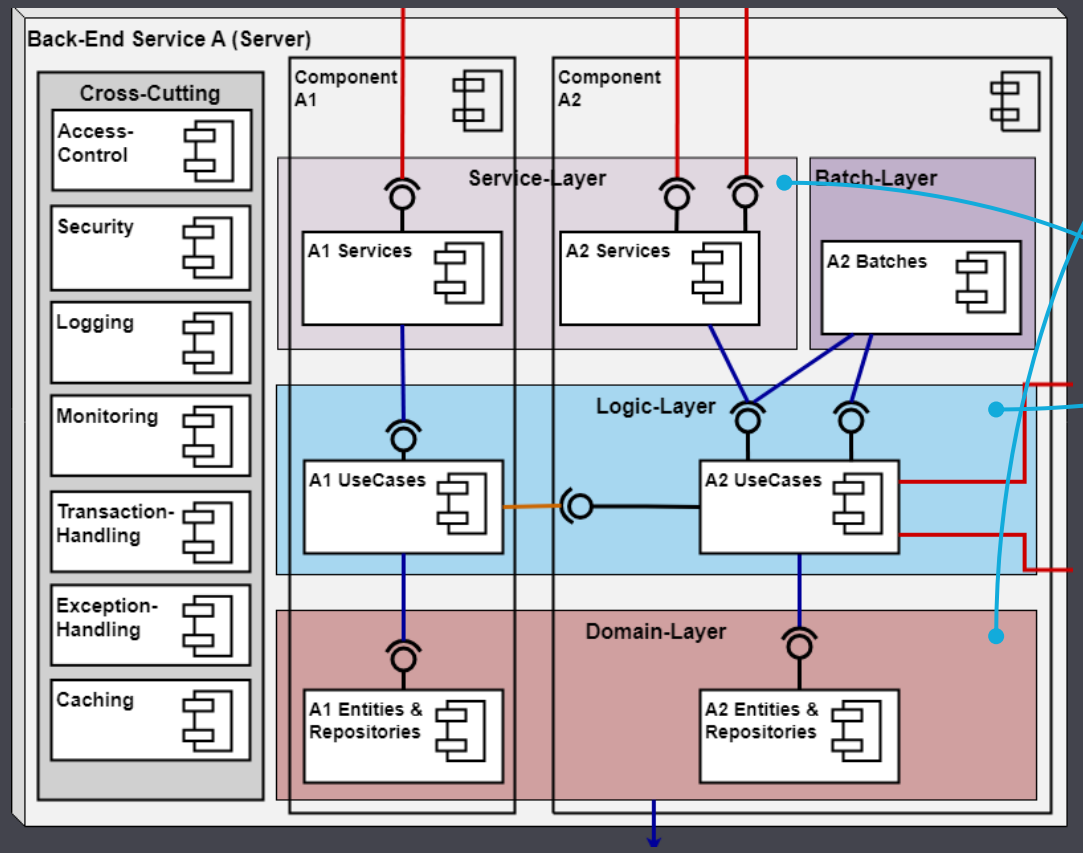
WHY LAYERED ARCHITECTURE?



LAYERED ARCHITECTURE



LAYERED ARCHITECTURE IN PROJECT STRUCTURE




```
«root»
├─.«component»
│  ├─.domain
│  │  ├─.repository
│  │  │  ├─.«BusinessObject»Repository
│  │  └─.model
│  │     └─.«BusinessObject»Entity
│  ├─.logic
│  │  ├─«BusinessObject»Validator
│  │  ├─«BusinessObject»EventsEmitter
│  │  └─.Uc«Operation»«BusinessObject»
│  └─.service
│     └─.rest
│        └─.v1
│           ├─.«Component»RestService
│           ├─.mapper
│           │  └─.«BusinessObject»Mapper
│           └─.model
│              └─.«BusinessObject»Dto
```

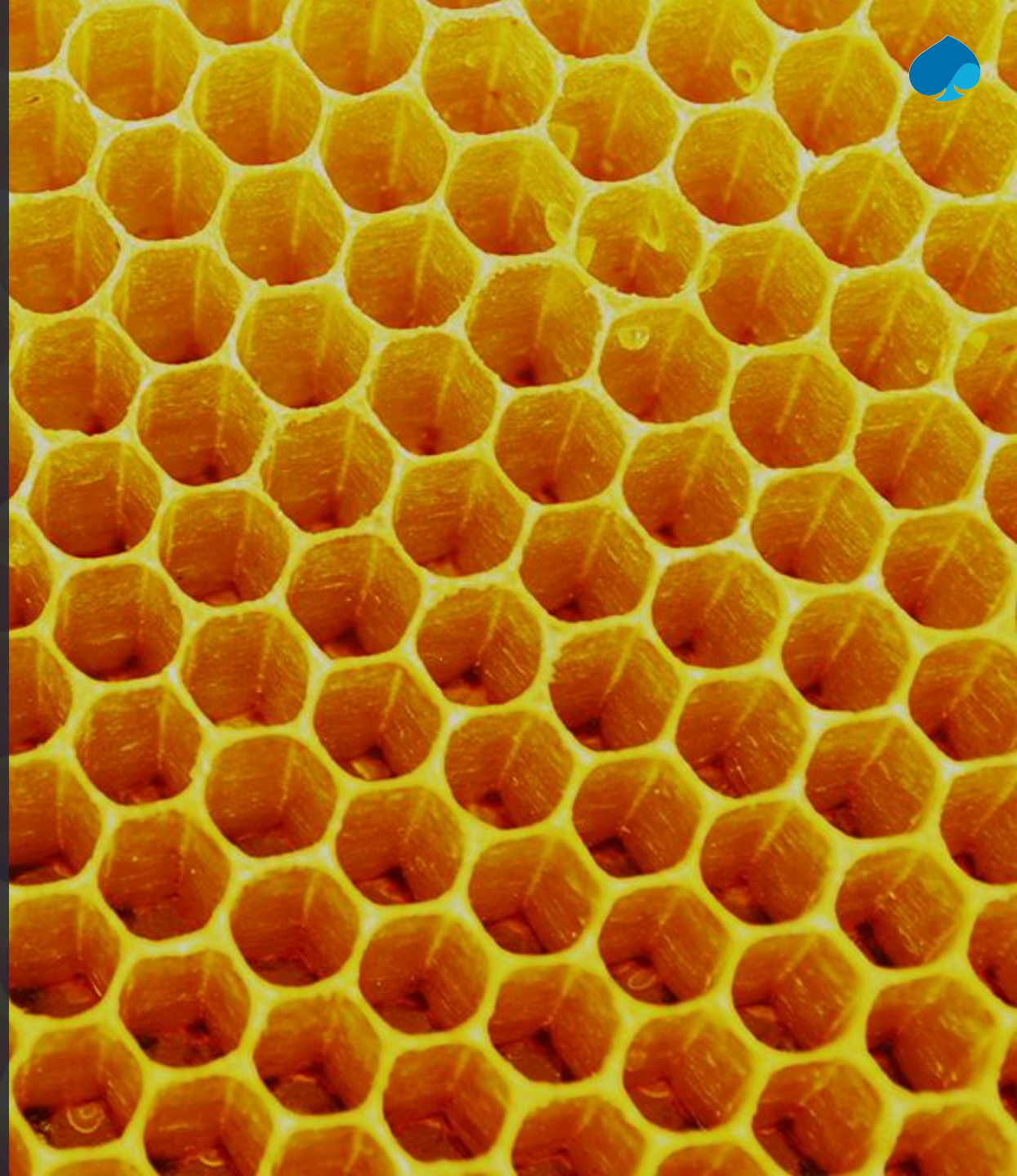
CHALLENGES WITH LAYERED ARCHITECTURE

Interdependency of Layers

Code creeping across layers

Q. How can we address these challenges?

HEXAGONAL ARCHITECTURE



WHO AM I?

- 13+ years of IT Experience
- Role – Technical Architect
- Aspiring L1 Architect
- Expertise in
  - System Design
  - Microservices

Sneha Sarmokadam

HEXAGONAL ARCHITECTURE?

What is Hexagonal Architecture?

- Introduced by Alistair Cockburn in 2005
- Known as “Port and Adapter” architecture
- Hexagonal architecture is a model of designing software applications around domain logic to isolate it from external factors

Why we need it?

- Avoids entanglement and logic leakage between layers and external components

HEXAGONAL ARCHITECTURE

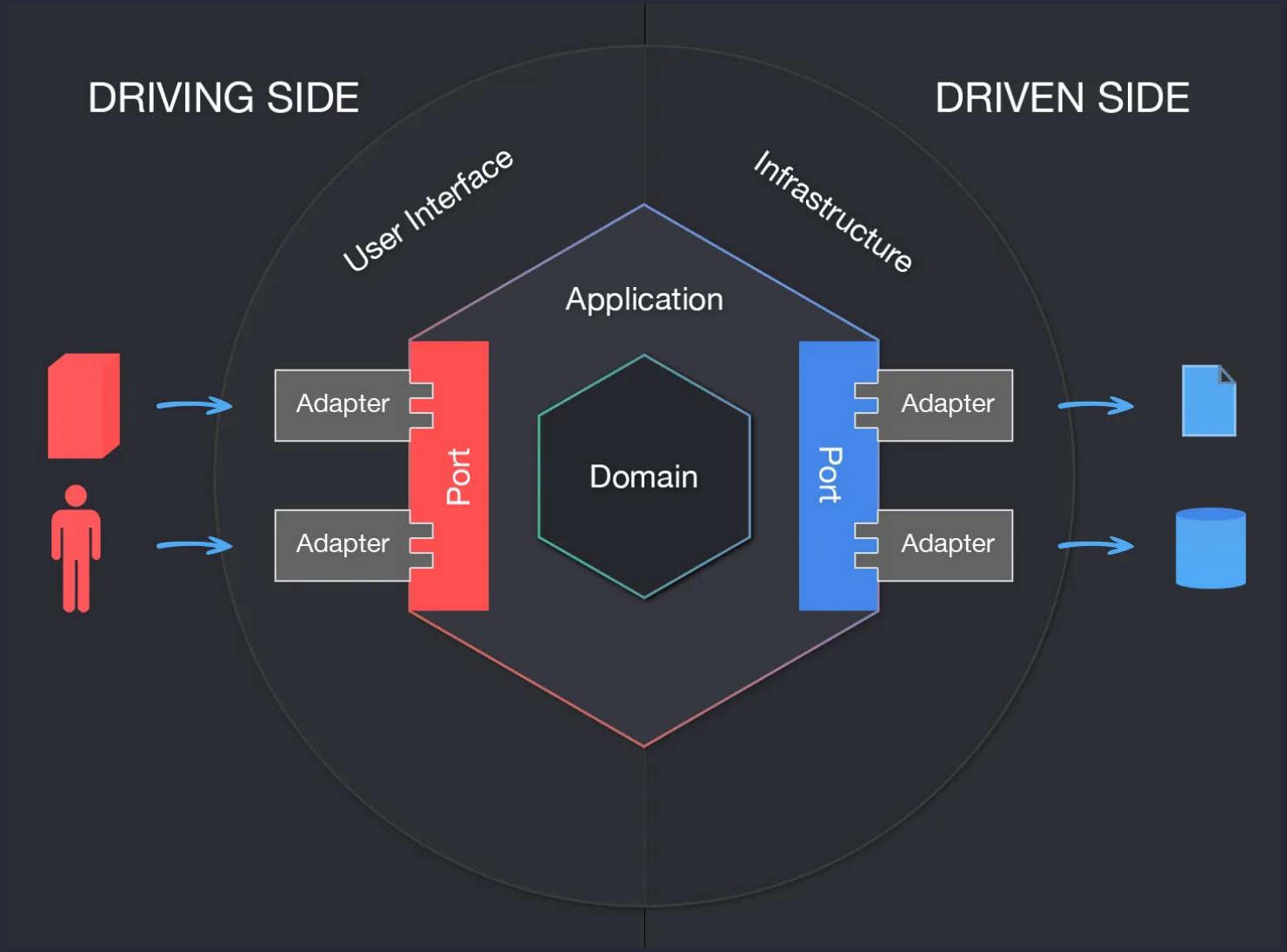


Image credit:https://medium.com/ssense-tech/hexagonal-architecture-there-are-always-two-sides-to-every-story-bc0780ed7d9c

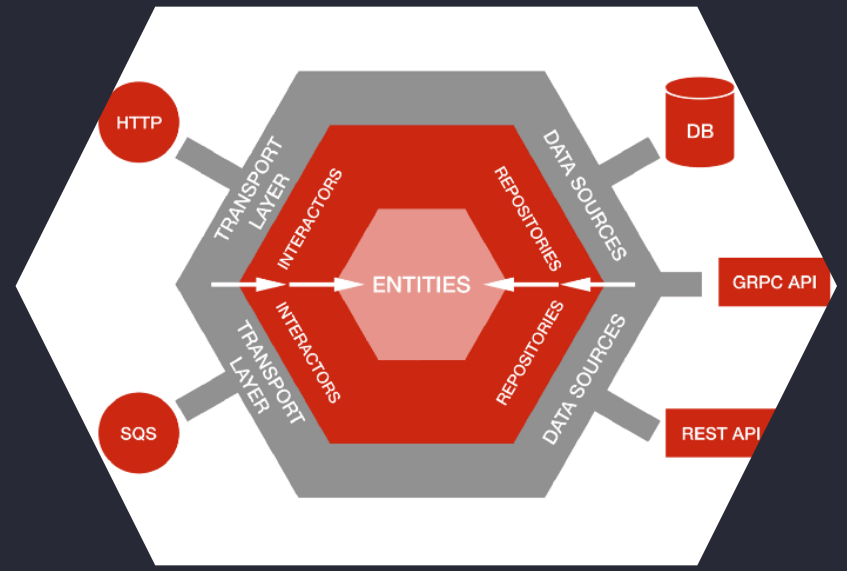


Image credit: https://netflixtechblog.com/ready-for-changes-with-hexagonal-architecture-b315ec967749

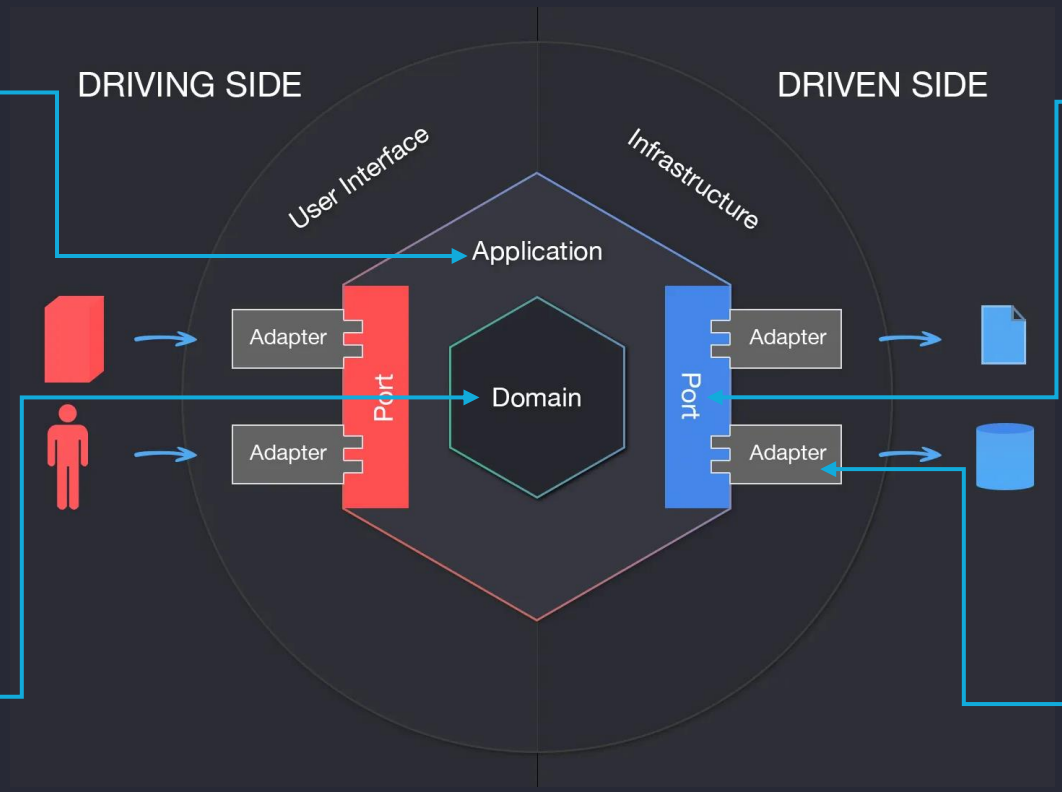
HEXAGONAL ARCHITECTURE

APPLICATION

Contains a core business logic such as use cases, ports

DOMAIN

Contains a core business logic such as model classes



PORT

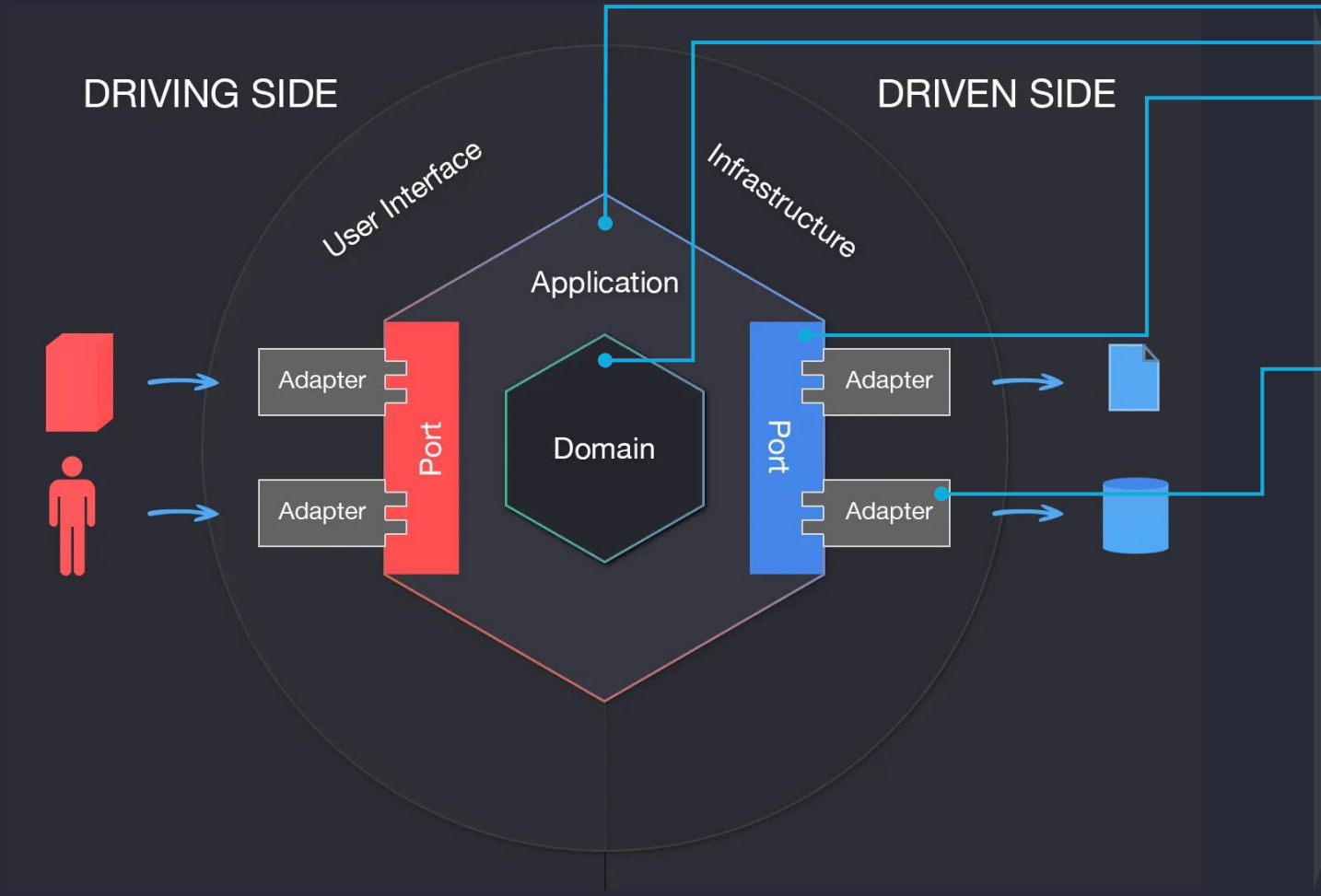
Interfaces defined by application to interact with external world

ADAPTER

Implementation of ports

Image credit:https://medium.com/ssense-tech/hexagonal-architecture-there-are-always-two-sides-to-every-story-bc0780ed7d9c

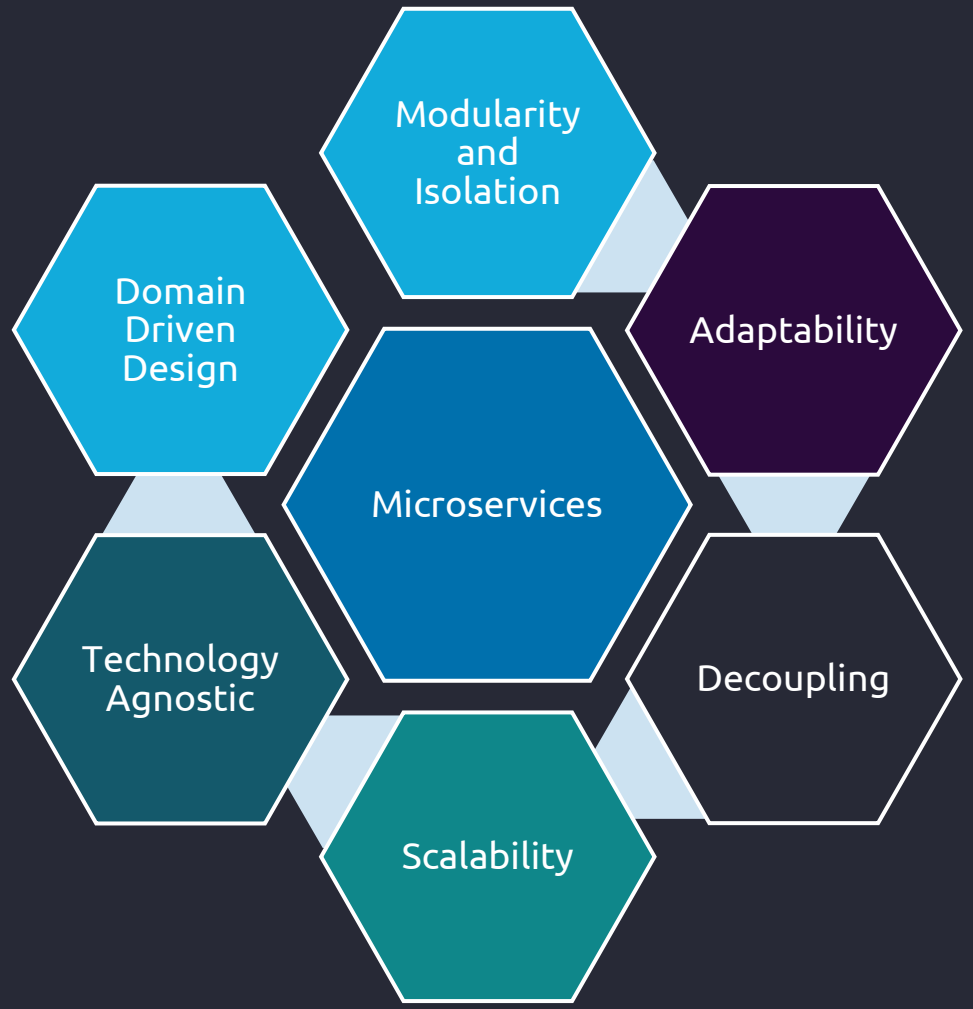
HEXAGONAL ARCHITECTURE IN PROJECT STRUCTURE




```
«root»
├──.application [core of application]
|   ├──.domain
|   |   ├──.BookEntity [model classes]
|   ├──.ports [interfaces]
|   |   └──.in
|   |   |   └──.«Business»Port
|   |   └──.out
|   |   |   └──.«Business»Port
|   └──.usecase
|   |   └──.«BusinessObjective»Uc
|   |
|   ├──.adapter [In adapters]
|   |   ├──.in
|   |   |   ├──.Job
|   |   |   |   ├──.«PortName»«Type»Adapter
|   |   |   ├──.queue [interfaces]
|   |   |   |   └──.«PortName»«Type»Adapter
|   |   |   └──.rest
|   |   |   |   └──.«PortName»«Type»Adapter
|   |   ├──.out
|   |   |   ├──.queue
|   |   |   |   ├──.«PortName»«Type»Adapter
|   |   |   ├──.db [interfaces]
|   |   |   |   └──.«PortName»«Type»Adapter
|   |   |   └──.file
|   |   |   |   └──.«PortName»«Type»Adapter
└──.infrastructure
    ├──.config
    └──.Logging
```

Image credit:https://medium.com/ssense-tech/hexagonal-architecture-there-are-always-two-sides-to-every-story-bc0780ed7d9c

WHY HEXAGONAL IS MORE SUITABLE FOR MICROSERVICES?

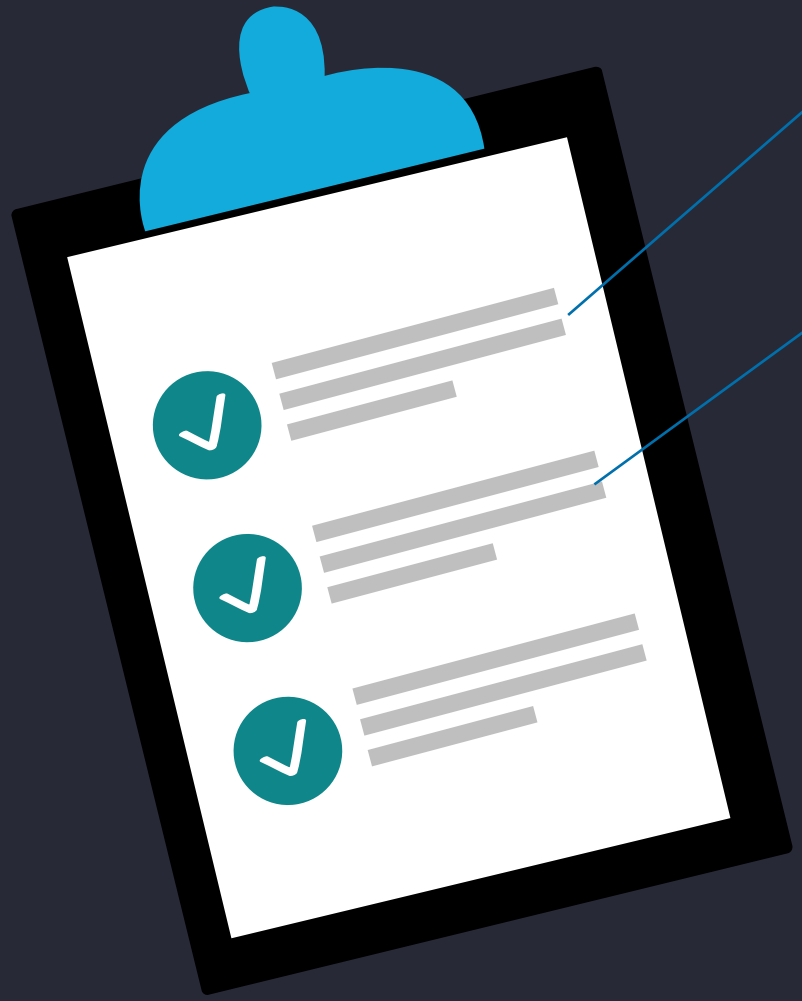



LAYERED
VS
HEXAGONAL
ARCHITECTURE

LAYERED VS HEXAGONAL ARCHITECTURE

Feature	Layered architecture	Hexagonal architecture
Structure	A set of stacked layers	A core domain surrounded by adapters
Dependencies ★	Layers depend on each other	The core domain does not depend on the adapters
Flexibility	Difficult to change the structure of the layers	Easier to change the adapters
Testability	May become difficult to test the interactions between layers	Easier to test the core domain
Maintainability	May become difficult to maintain as the system grows	Easier to maintain as the system grows
Learn	Simple and easy to learn	Relatively higher learning curve
Suitable for	Simple and small application - The requirements for the project are: • stable and well-defined • need to build system faster	A system that needs to be able to interact with a variety of external systems A system that needs to be able to adapt to changes in the environment

SUMMARY



The best choice of architectural pattern will depend on the specific needs of the system.

Below are few pointers to decide which architectural pattern to use:

- Size & Complexity of the system
- External interfaces
- Non-functional requirements (like performance, scalability, etc)
- Customer preference, team's experience
- Time & Budget constraints

No architecture pattern can fulfill all system requirements, rather we find optimal patterns based on key requirements & principals

Capgemini

THANK YOU

Quiz?

(https://kahoot.it/)

Feedback

HTTPS://FORMS.OFFICE.COM/E/3VTQBB1AH0