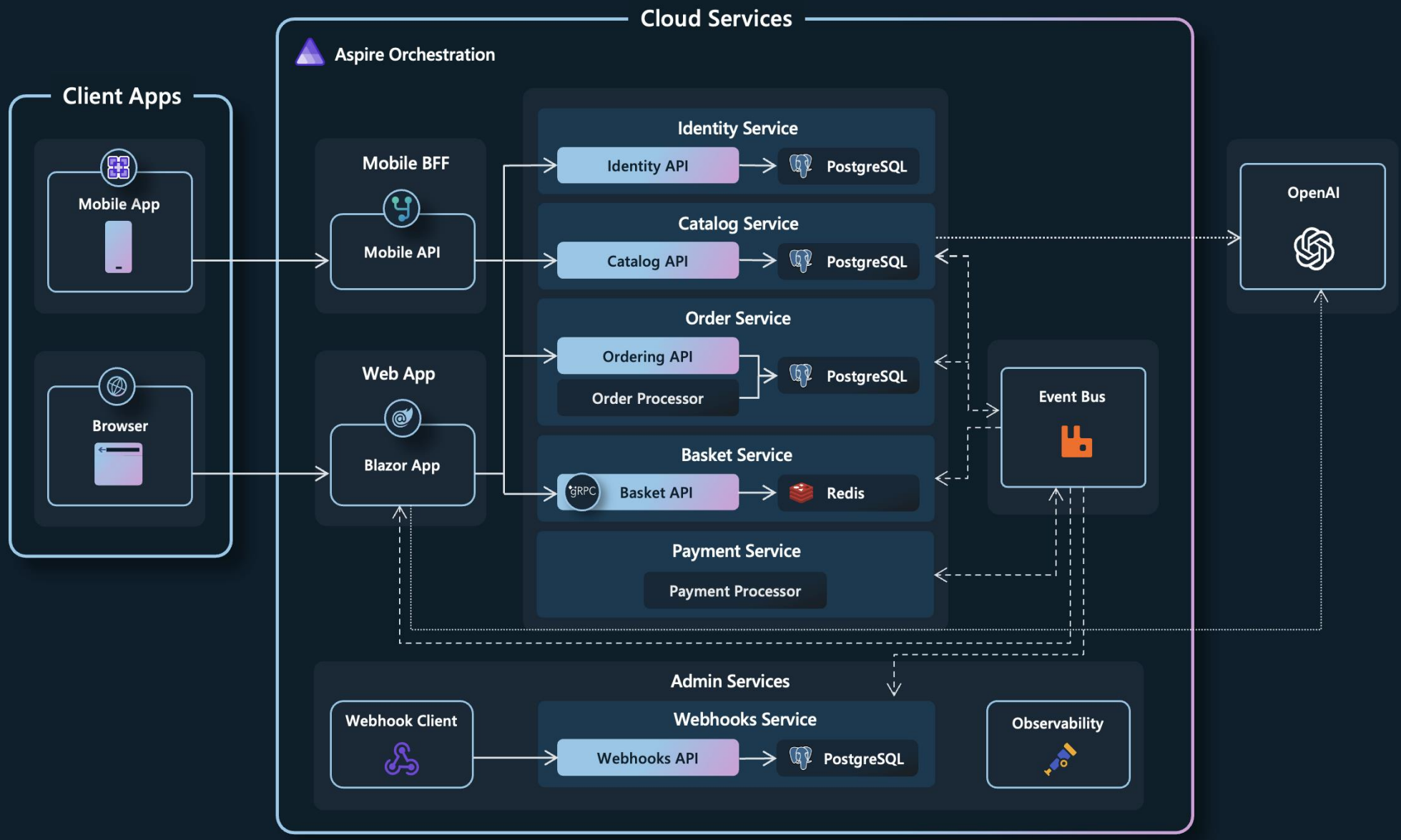
Розробка додатків на платформі .NET

Навчально-науковий інститут фізико-технічних та комп’ютерних наук

Кафедра комп’ютерних наук

Томка Ю.Я.

Чому присвячений курс?



Full-stack web development мовою C# «cloud-native» застосунків на мікросервісній архітектурі

Проблеми, що виникають у рамках «подібних курсів» та які шляхи їх подолання

Використання точкових задач та завдань, що не надають повноцінної картини та структури реального IT-проекту для студента

Важкість контролю викладачем швидкості та якості виконання сформованого перед студентом завдання

Викладач не слідкує за тенденціями IT-ринку. Не розуміє, що актуально сьогодні, а що - ні

Побудувати курс на індивідуальних (проте подібних) проектах, розбивши його на частини. Кожна окрема частина може розглядатись як окрема задача. У своїй сукупності вони формують повноцінний проект

Максимально залучити систему контролю версій та інструментарію CI/CD у зав'язці на сучасні методології розробки програмного забезпечення

Неперервний моніторинг актуальних інструментів, фреймворків, порівняння із останніх із аналогами. Відбудувати логіку побудови та написання проекту із залученням сучасного інструментарію

Тотальне ігнорування необхідності покриття розробляємого проекту юніт- та інтеграційними тестами	Написання пакету юніт- та інтеграційних тестів для навчальних проектів із відповідним аналізом проценту покриття коду проекту
Ігнорування проблеми вибору архітектури веб-додатку. Класично сформований підхід, що обмежується вибором MVC та поверхневе розуміння даної задачі	Надати можливість для студента здійснити вибір архітектури веб-додатку в рамках якої він буде вести розробку (за основу взяти монолітні рішення – багатошарова/чиста архітектура)
Як правило, курс завершується класичним екзаменом, відірваним від реалій IT-сфери	Екзамен проводити у формі демонстрації/захисту проектів із залученням всіх зацікавлених. Акцентуалізувати увагу на проблемах із якими стикався студент та шляхах їх рішення

Неможливість використання групових проектів із розділенням тасків між студентами через різну вмотивованість

Оцінювання протягом курсу здійснюється на основі застарілих методиках, що базуються на факті «зробив/не зробив»

Захист потрібен тільки викладачу і студенту «для оцінки»

Формування індивідуальних проектів (персональних) із схожою структурою та бізнес-логіками. Ідентичні завдання, як правило, призводять до ідентичних проблем, що підштовхують до роботи в команді

Побудувати систему оцінювання, яка включала можливість оцінювати не тільки факт виконання але й аналіз проблеми, підходи до рішення проблеми, генерування ідей, апробацію декількох шляхів рішення, роботу в команді, тайм-менеджмент, креативність

Залучення представників IT-сфери

На що спираємось з попередніх (раніше прослуханих) курсів

Необхідно вести постійний моніторинг структури інших курсів. Чітко розуміти, що буде використовуватись в рамках даного курсу.

Об’єктно-орієнтоване програмування

- Основи C#
- SOLID-принципи
- GOF - патерни
- EF CORE
- LINQ
- ADO.NET + DAPPER

Організація баз даних та знань

- MSSQL server
- SQL

Алгоритмізація та програмування

- Базові алгоритмічні конструкції

Веб-технології та веб-дизайн

- HTML
- CSS
- SASS/LESS
- BOOTSTRAP
- Основи JS

Перед початком курсу необхідно наголосити, що саме студент повинен повторити для успішного старту.

Спосіб інформування – лист у вигляді майнд-мапи курсу

ФОРМУВАННЯ ПРОБЛЕМИ ТА ТЕХНІЧНОГО ЗАВДАННЯ

Викладач формує орієнтовану тематику проектів. При цьому їхня бізнес-логіка повинна бути приблизно однаковою із невеликими відмінностями в розрізі предметної області

Однакова структура задач бізнес-логіки підштовхує до обговорення між студентами шляхів розв'язку ідентичних проблем

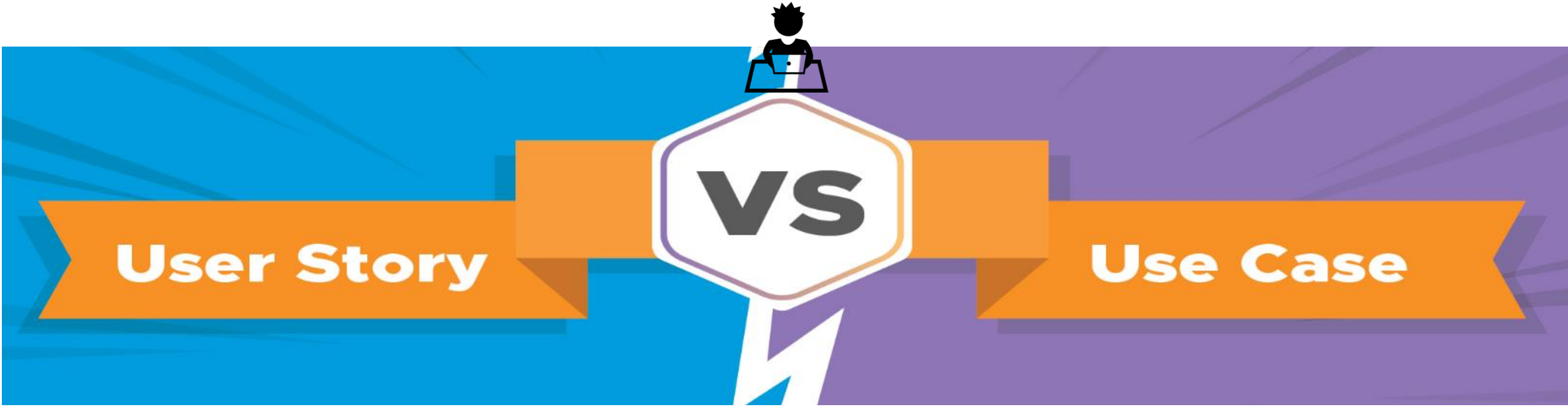
Підвищується рівень командної роботи

Росте вмотивованість студента через його роботу над цікавим для нього проектом

Може обрати тематику із наданого викладачем списку

Пропонує власну тематику і після обговорення з викладачем вона затверджується до виконання

Як показує практика у студента не формується чітке розуміння відповідного інструментарію для опису того функціоналу який повинен надаватися розробляємим додатком



Глибинний аналіз та порівняння User Stories із Use Cases є необхідною умовою саме на старті обговорення проекту та формування технічного завдання в розрізі практичної точки зору

За основу взяти базові принципи та ідеї Scrum

В якості платформи для системи контролю версій, керування проектом та розв'язку задач Continuous Integration / Continuous Deployment використати Github або Azure DevOps

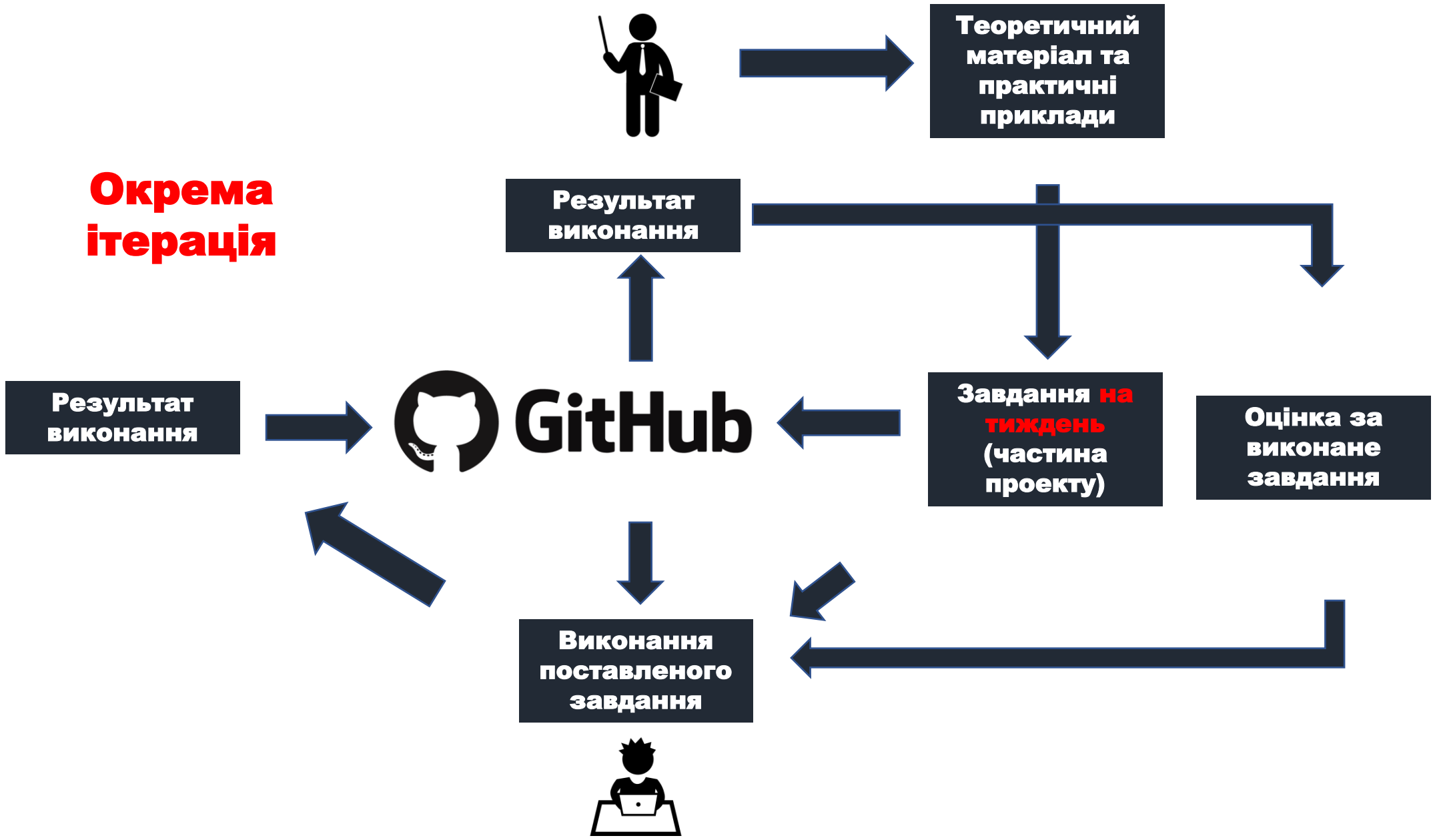
GitHub

Azure DevOps

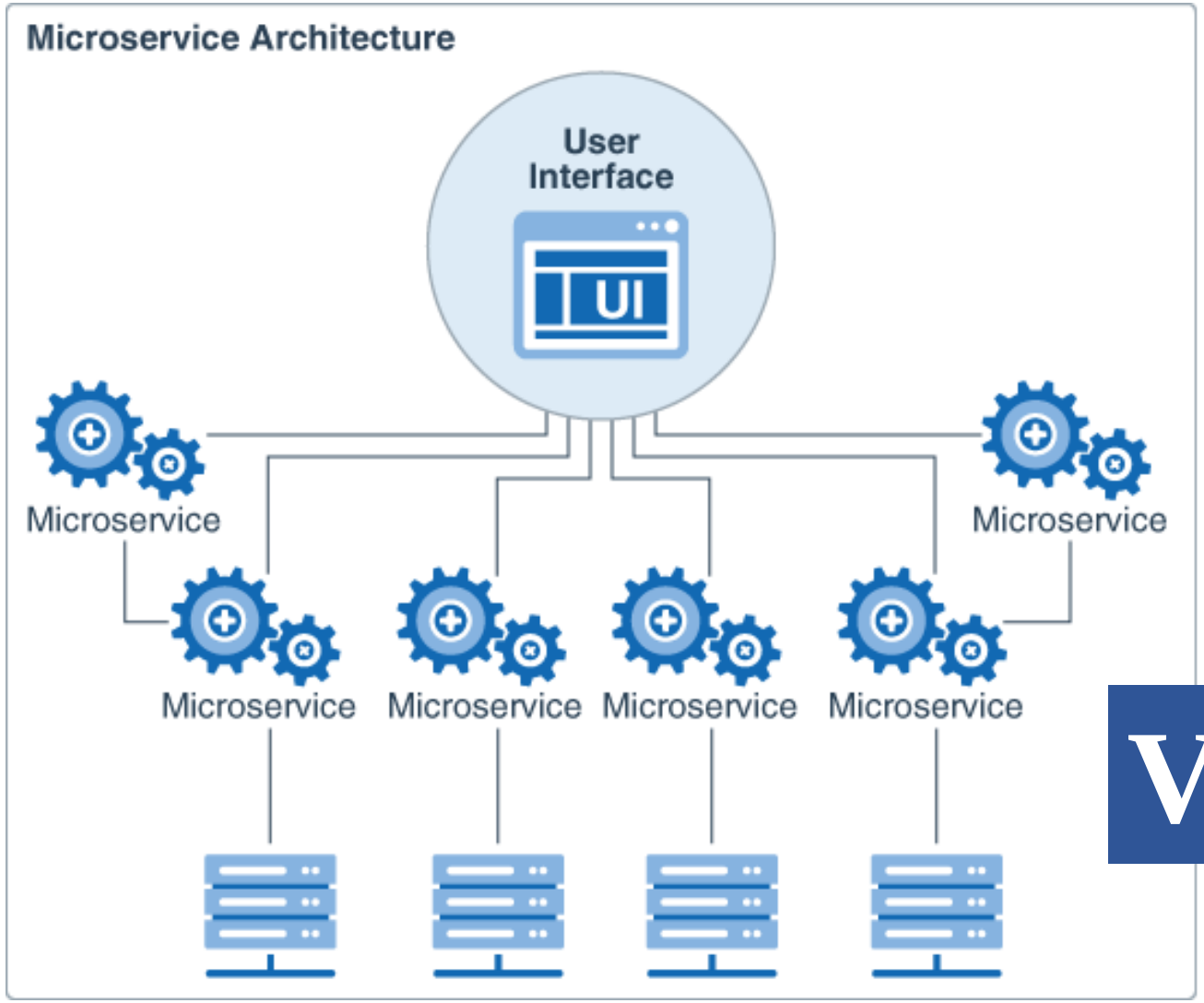
Контроль виконання проекту викладач здійснює на основі платформи Github або Azure DevOps. Також з її допомогою формує для студента відповідні завдання на тиждень

Як саме контролюємо виконання завдань на які розбитий проект?

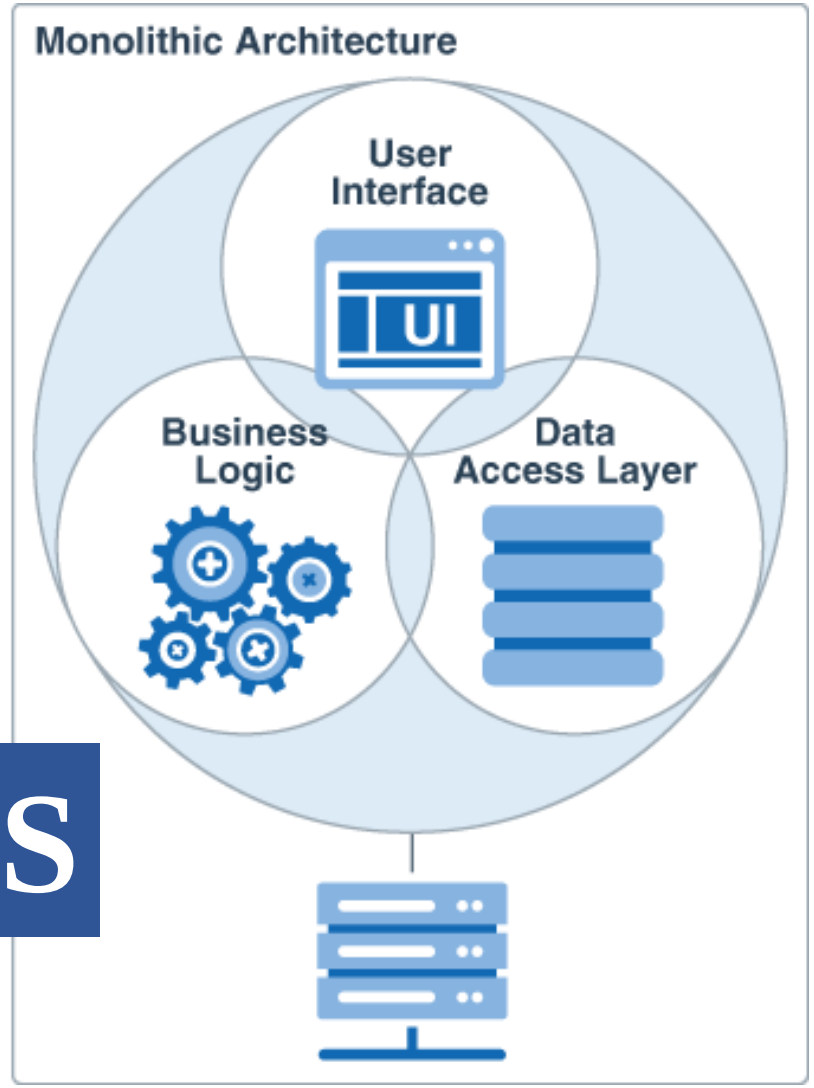
15 тижнів = 15 практичних занять



Питання архітектури та стеку технологій



VS



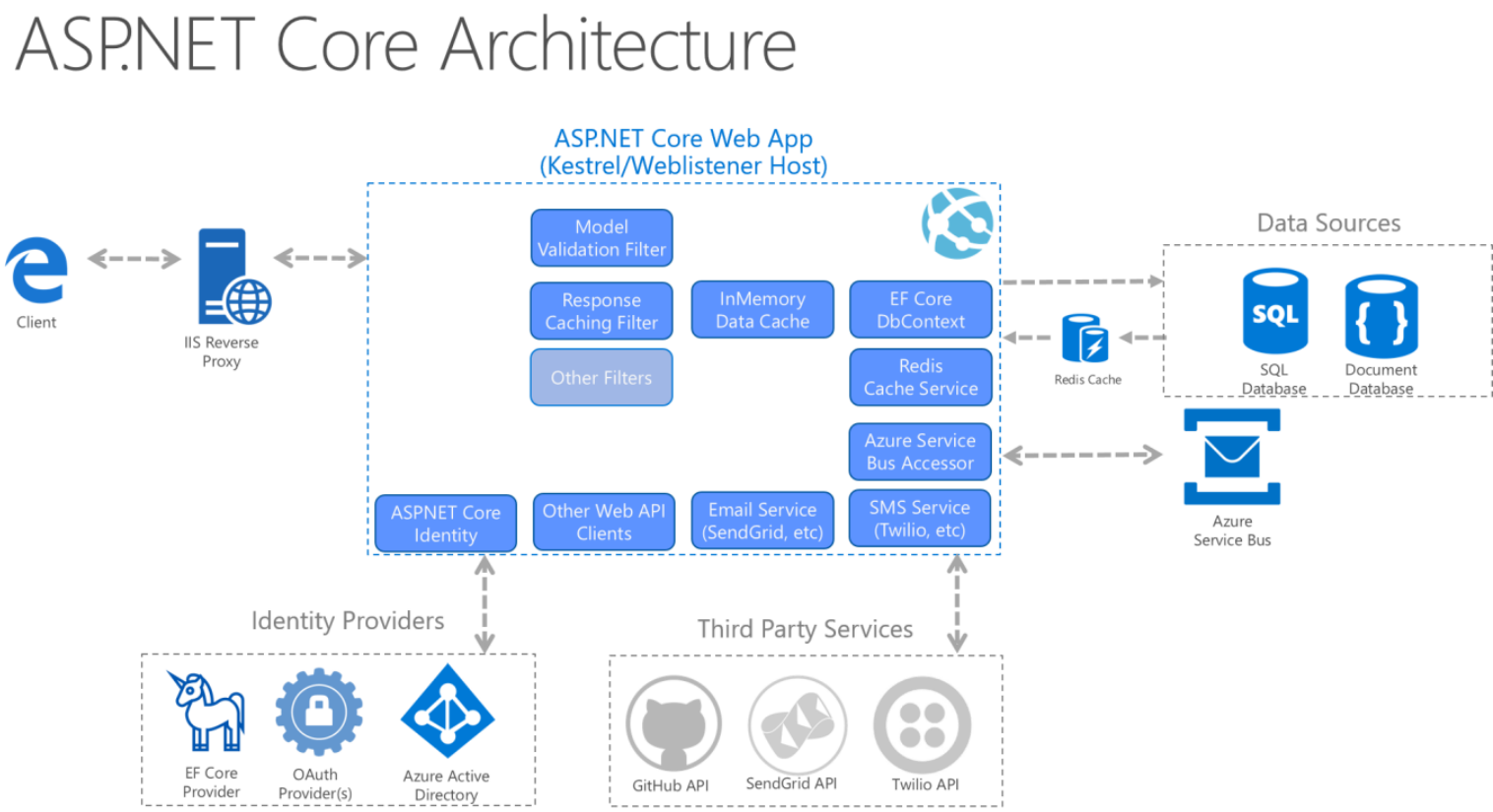
Зосередження уваги на більш поширених архітектурних рішеннях, які використовуються при розробці веб-додатків. Зосередження уваги на тому, що вибір архітектури може регламентувати вибір стеку технологій та відповідних паттернів

Що вивчаємо та що використовуємо при написанні проекту?

(схематичне представлення в розрізі вивчаэмих тем)

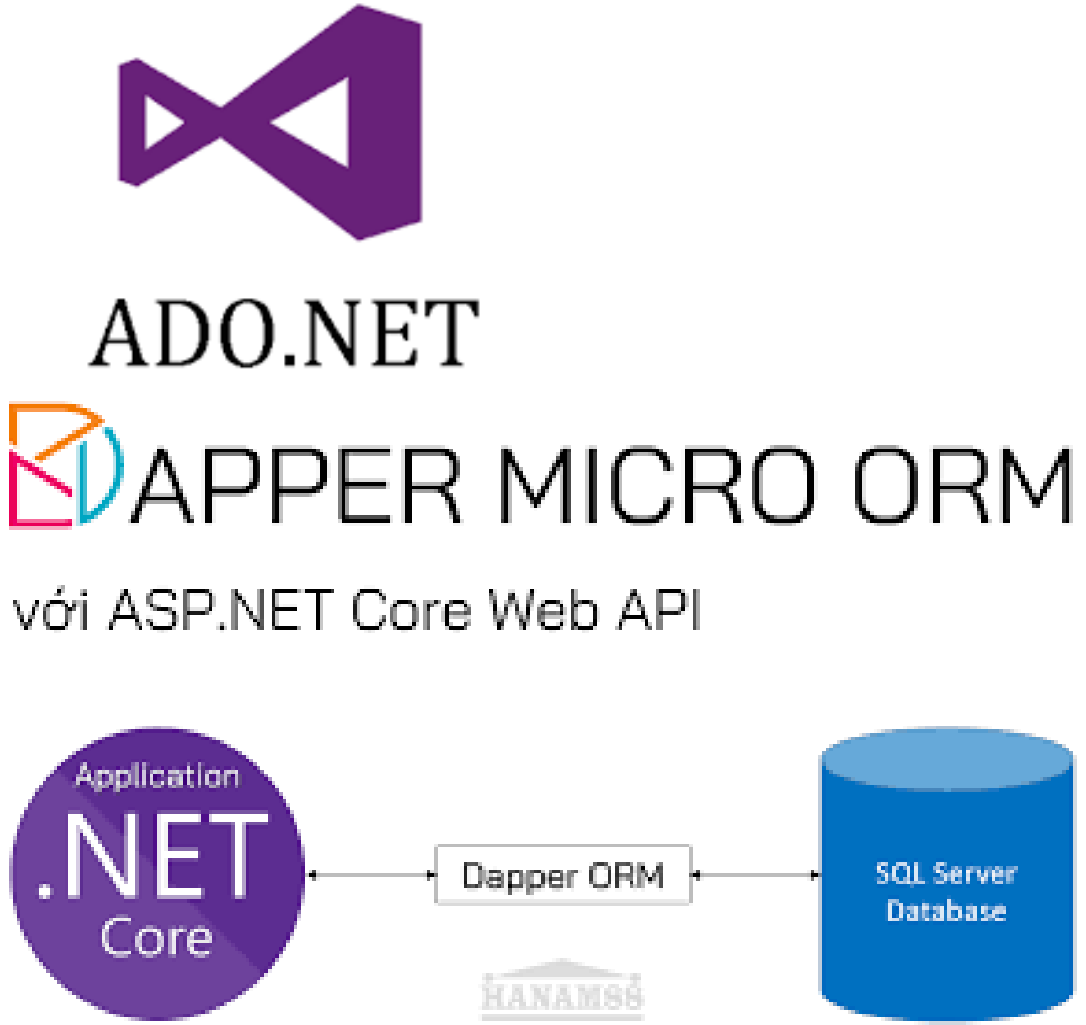
Тема 1. Вступ та основи ASP.NET Core. Принципи побудови БД для мікросервісів

- Загальна структура проекту;
- Основи конфігурації
- Dependency injection;
- Логіювання;
- Маршрутизація;
- Представлення та компоновка;
- Контролери та дії



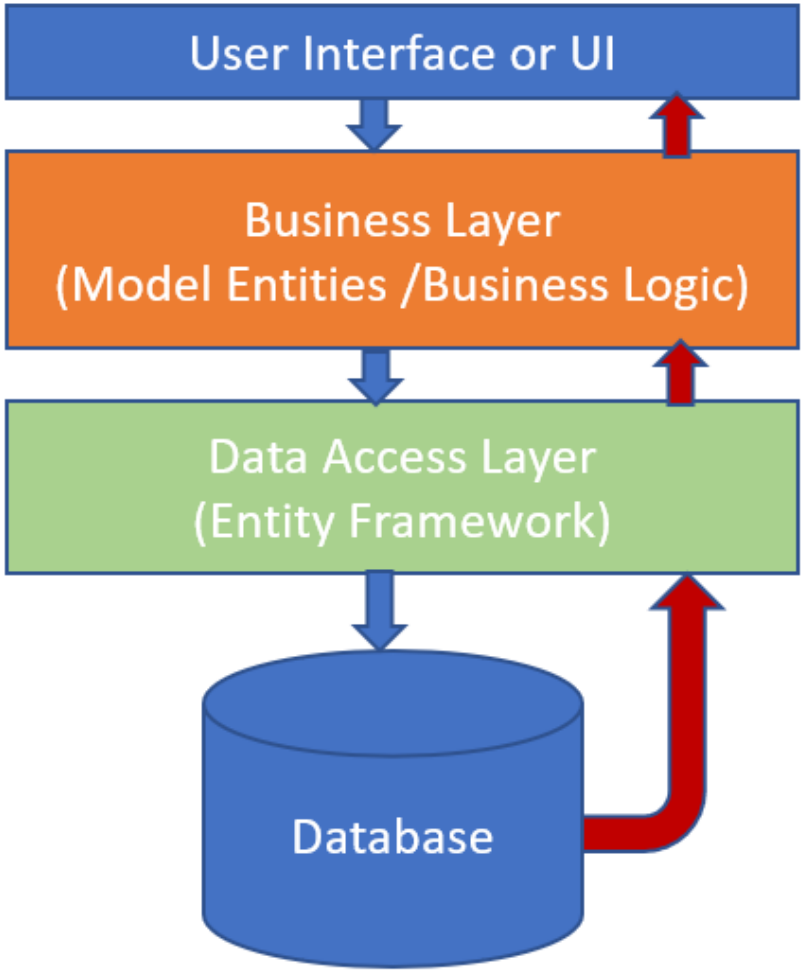
Тема 2. Багатошарова архітектура на основі ADO.NET/Dapper.

- Поняття багатошарової архітектури;
- DAL;
- BLL;
- WEB;
- CRUD – операції;
- Підєднаний режим роботи ADO.NET;
- SQL;
- Dapper



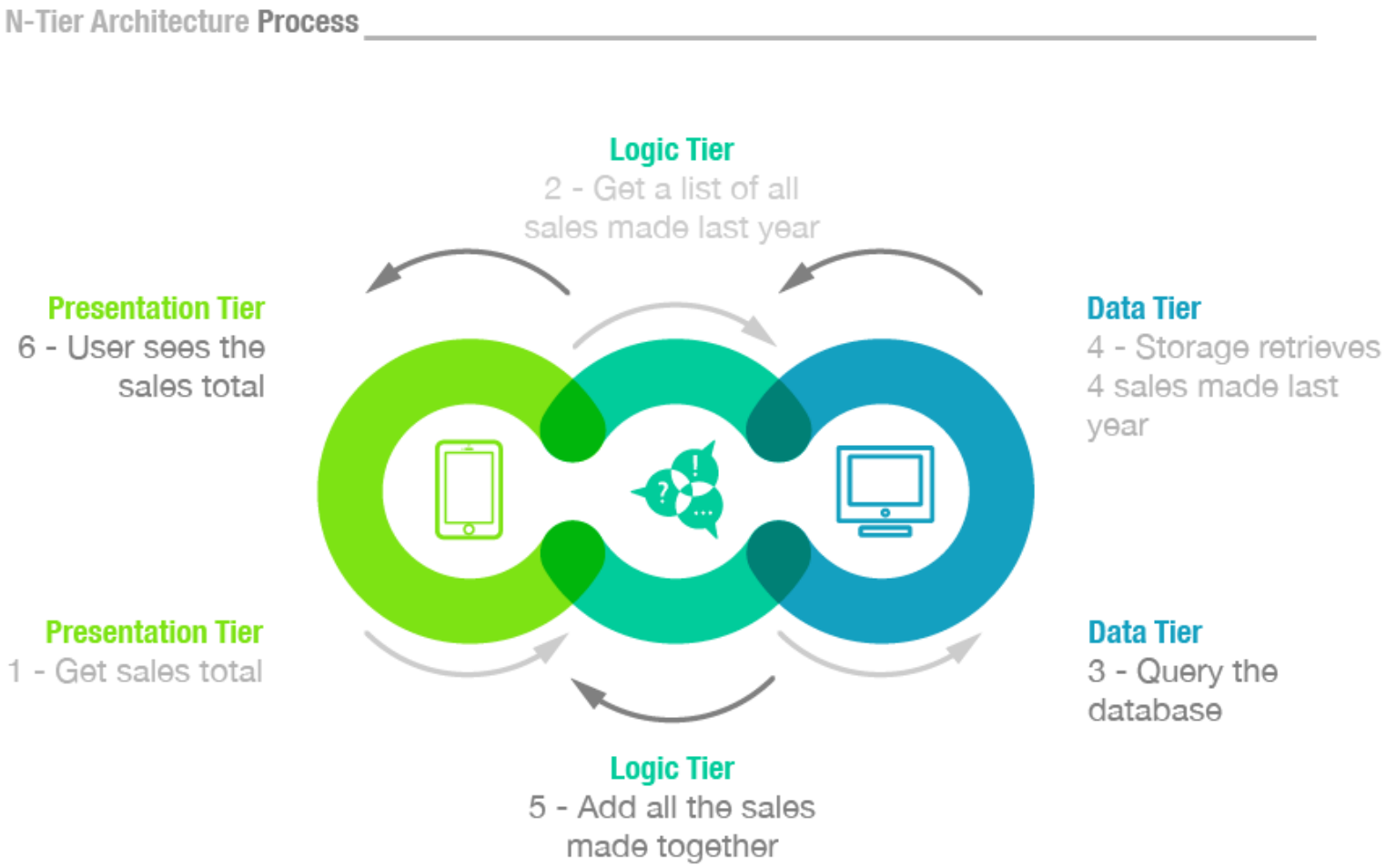
Тема 3. Багатошарова архітектура на основі Entity Framework Core. Data Access Layer. LINQ to Entities. Паттерни Generic Repository/Unit of Work

- Linq to Objects;
- Linq to Entities;
- POCO Objects;
- Fluent API;
- Generic Repository;
- UOW;
- DbSet;
- DbContext;
- Робота із міграціями



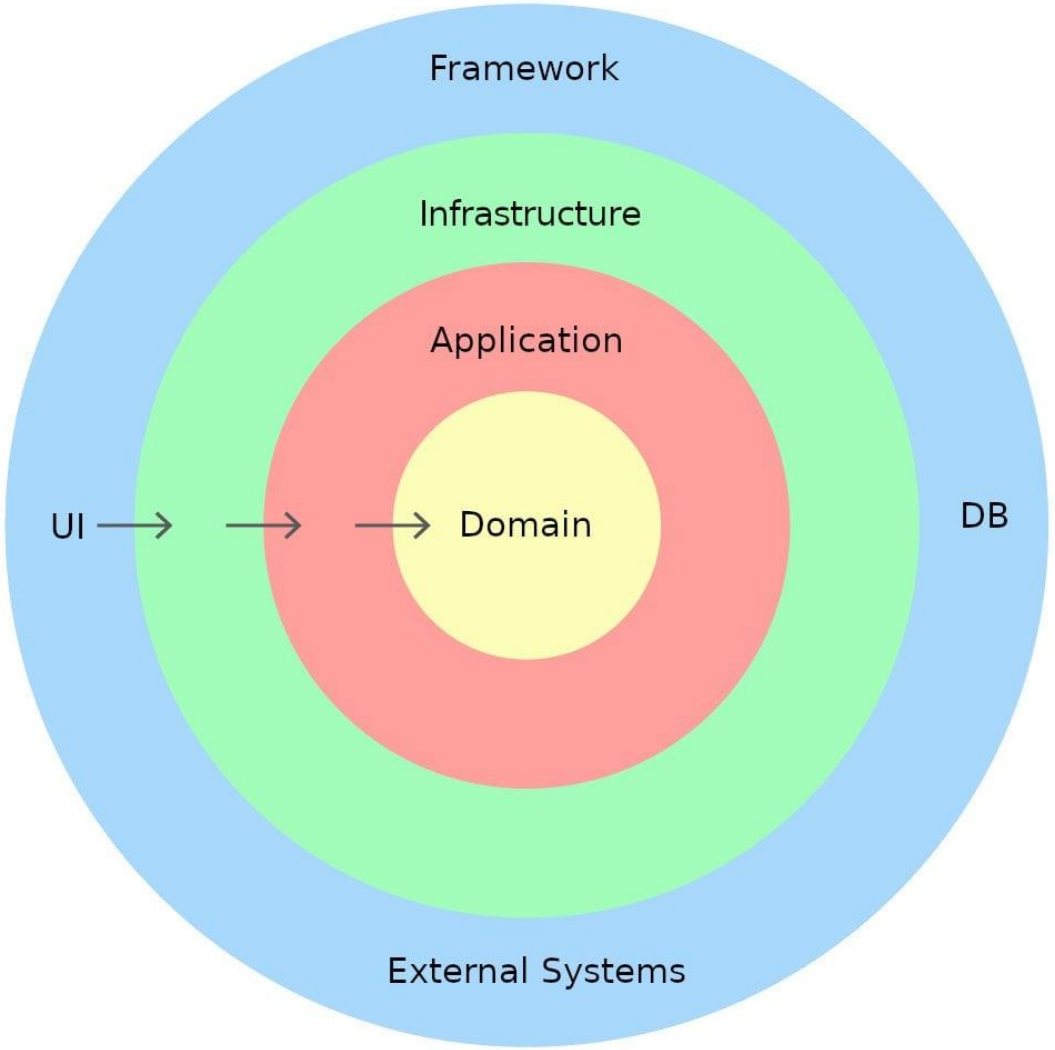
Тема 4. Багатошарова архітектура на основі Entity Framework Core. Business Logic Layer та Web-Layer

- User Stories;
- Use Cases;
- Сервіса/Менеджери;
- DTO;
- Мапінг даних;
- Dependency Injection;
- AutoMapper;
- Маршрутизація;



Тема 5. Чиста архітектура. CQRS. MediatR.

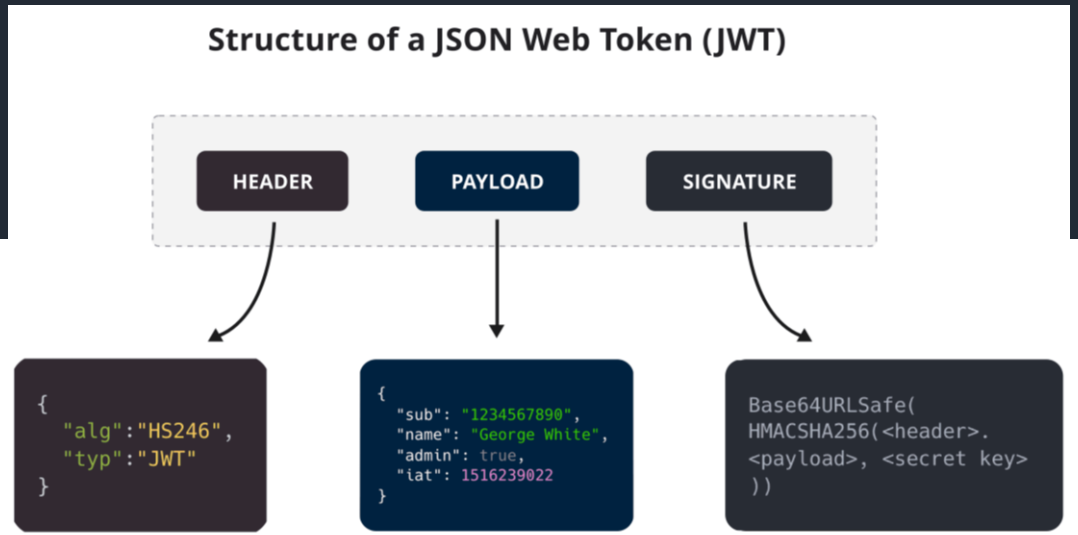
- **Детальний аналіз кожного шару;**
- **Загальна структура проекту;**
- **Value Objects;**
- **CQRS;**
- **MediatR;**
- **Fluent Api;**



Тема 6. Питання аутентифікації та авторизації. ASP.NET Core Identity. JWT-токен

- Вступ до ASP.NET Core Identity
- Основні класи в ASP.NET Core Identity
- Додавання Identity до проекту з нуля
- Реєстрація та створення користувачів у Identity
- Авторизація користувачів у Identity
- Керування користувачами
- Зміна пароля
- Валідація пароля
- Валідація користувача
- Управління ролями
- Ініціалізація БД ролями та користувачами

- Аутентифікація за допомогою JWT-токенів
- Авторизація за допомогою JWT-токенів у клієнті JavaScript
- Аутентифікація за допомогою куки
- HttpContext.User, ClaimPrincipal та ClaimsIdentity
- ClaimPrincipal та об'єкти Claim
- Авторизація за ролями
- Авторизація на основі Claims



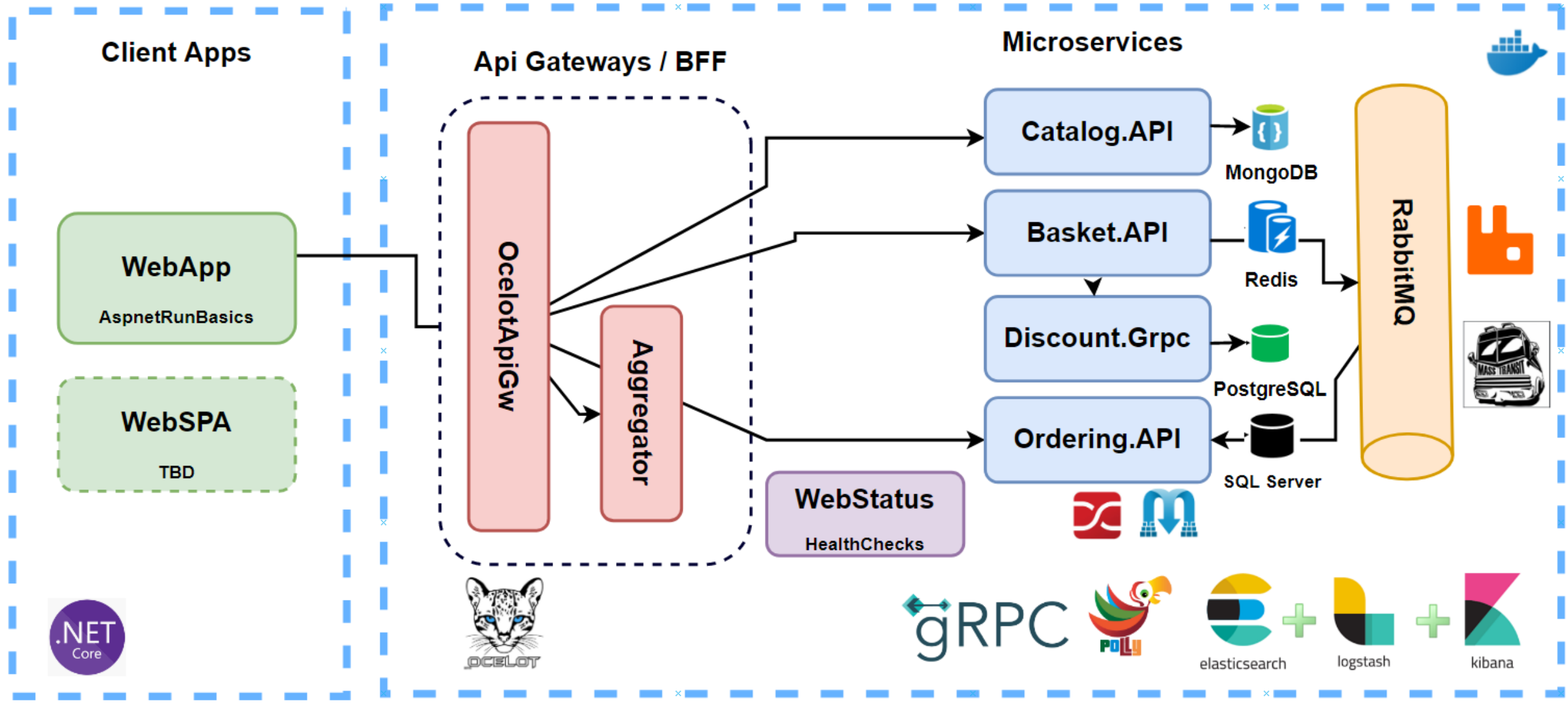
Тема 7. Клієнт-серверні застосунки на плаформі Asp.Net та GraphQL

- Building a basic GraphQL server API.
- Controlling nullability.
- Understanding GraphQL query execution and DataLoader.
- GraphQL schema design approaches.
- Understanding middleware.
- Adding complex filter capabilities.
- Adding real-time functionality with subscriptions.
- Testing the GraphQL server.

GraphQL

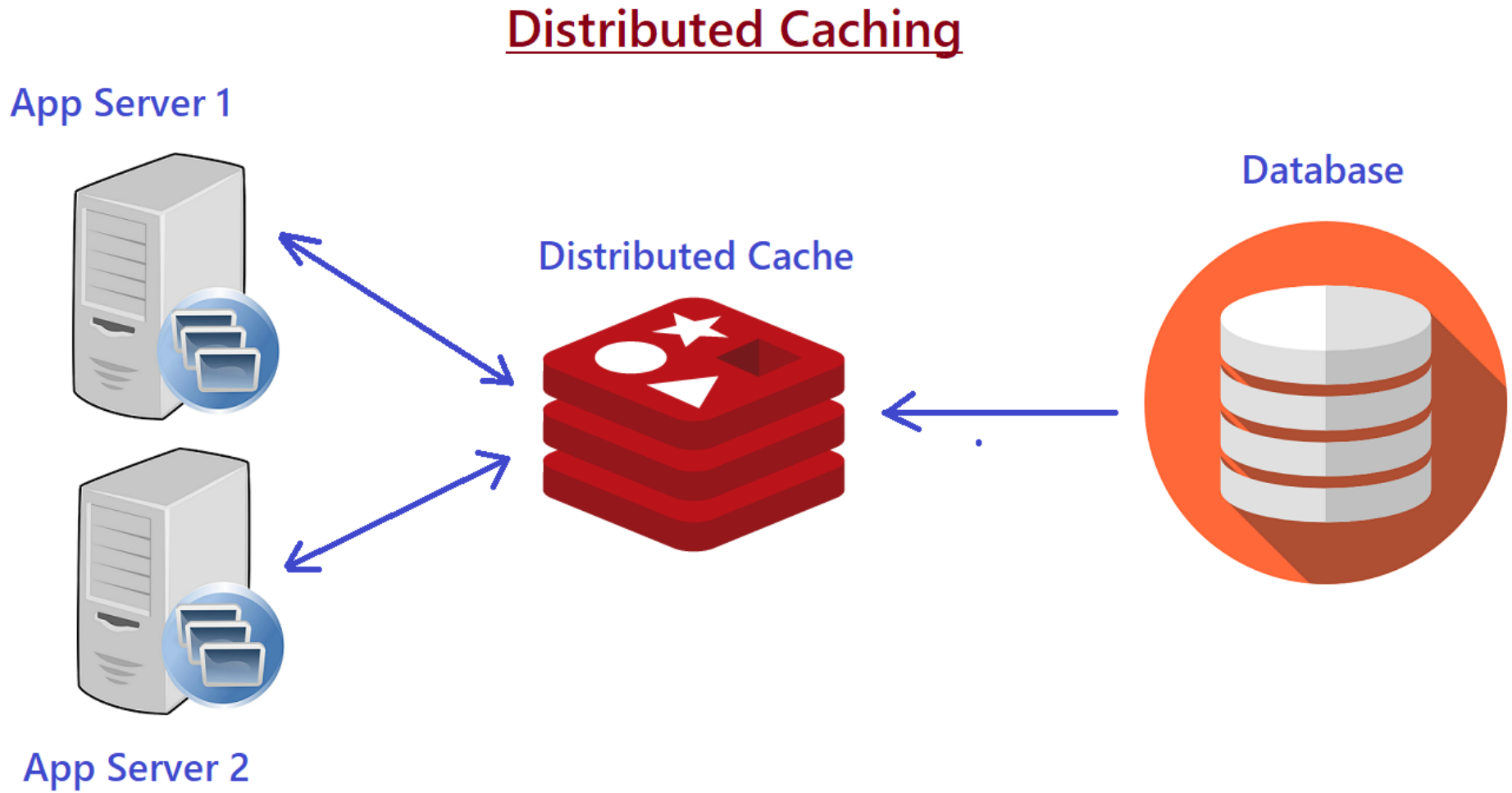
With Hot Chocolate

Тема 8. Microservice architecture. Основні поняття та концепції



Тема 9. Microservice architecture. In-memory & distributed (Redis) caching in Asp.net Core

- Data types
- Using Redis
- Managing Redis
- Redis Stack



Тема 10. Microservice architecture. Побудова Api Gateway із використанням Ocelot. Дизайн-патерн «Агрегатор»

- Routing
- Administration
- Rate Limiting
- Caching
- Headers Transformation
- HTTP Method Transformation
- Claims Transformation
- Logging
- Tracing
- Middleware Injection and Overrides
- Load Balancer
- Delegating Handlers
- Http Error Status Codes

Тема 11. Microservice architecture. Використання gRPC для побудови високопродуктивного міжмікросервісного зв'язку на платформі .Net

- Служби gRPC у ASP.NET Core
- Використання gRPC у програмах на основі браузера
- Аутентифікація та авторизація
- Перехоплювачі gRPC
- Оптимальні методи підвищення продуктивності
- Міжпроцесна взаємодія
- Клієнти та служби з підтримкою Code First
- Управління посиланнями protobuf за допомогою .NET gRPC
- Міграція з C-core на gRPC для .NET
- Навіщо виконувати міграцію WCF на ASP.NET Core gRPC
- Порівняння служб gRPC із API-інтерфейсами HTTP

Тема 13. Microservice architecture. Захист мікросервісів за допомогою Identity Server4, Oauth 2.0 та OpenId Connect під управлінням Ocelot Api Gateway

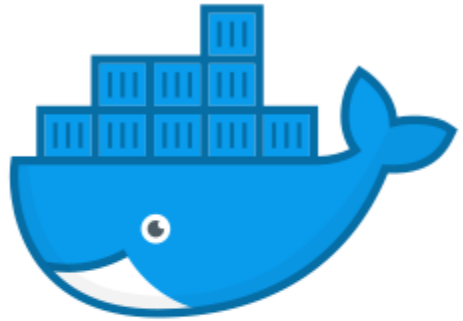
- Protecting an API using Client Credentials
- Interactive Applications with ASP.NET Core
- ASP.NET Core and API access
- Using EntityFramework Core for configuration and operational data
- Using ASP.NET Core Identity
- Building JavaScript client applications
- Building Blazor WASM client applications

Тема 14. Microservice architecture. Microservices Observability, Resilience, Monitoring on .Net

- Microservices Observability with Distributed Logging using ElastichSearch
- Microservices Resilience and Fault Tolerance with Appling Retry and Circuit–Breaker patterns using Polly
- Microservices Monitoring with Health Checks using WatchDog
- Microservices Tracing with OpenTelemetry using Zipkin

Тема 15. Основи розгортання .Net додатків на мікросервісній архітектурі у Kubernetes-кластер

- Основи Kubernetes
- Підняття простого K8s Cluster на Windows
- Підняття K8s Cluster в AWS - Amazon Web Services
- Підняття K8s Cluster у GCP - Google Cloud Platform
- Підняття K8s Cluster Для Навчання - Безкоштовно в Інтернеті
- Створення DockerImage з Dockerfile та Завантаження в DockerHub
- Головні Об'єкти K8s
- Створення та керування - Pods
- Створення та керування - Deployments
- Створення та керування - Services
- Створення та керування - Ingress Controllers
- Створення та управління - Helm Charts



docker

kubernetes

ДОДАТКОВО

BLAZOR

OPEN SOURCE .NET WEB FRAMEWORK

Основи

Blazor WebAssembly vs Blazor server-side

Порівння із JS-аналогами

MatBlazor

Material Design components for Blazor (2.6.2)

BlazorStrap

Bootstrap 4 Components for Blazor Framework

FULLY COMPATIBLE WITH BLAZOR WEBASSEMBLY 3.2.0 OFFICIAL RELEASE!

Telerik UI for Blazor

DevExpress UI Components for Blazor

DevExpress Blazor UI Component Library ships with over 25 native Blazor components (including a DataGrid, Pivot Grid, Scheduler, Chart, Data Editors, and Reporting) so you can design rich user experiences for both Blazor Server and Blazor WebAssembly using C#.

ДОДАТКОВО

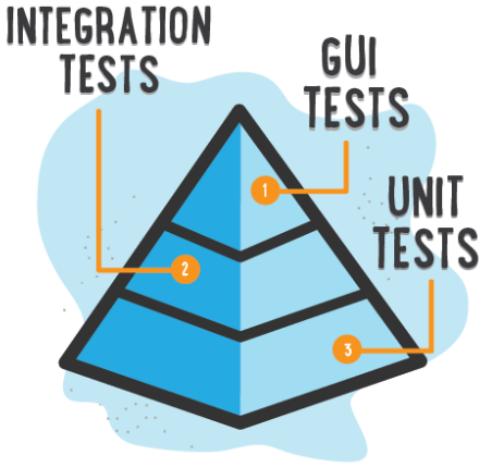
Основи інтеграційного тестування

Відмінності інтеграційного тесту від юніт-тесту

Базові тести зі стандартним WebApplicationFactory

Налаштування клієнту із WebHostBuilder

Робота зі службами імітації



КЛАСИЧНИЙ ФОРМАТ ДЕМО В РАМКАХ ЕКЗАМЕНУ

із

ЗАЛУЧЕННЯМ ПРЕДСТАВНИКІВ ІТ-СФЕРИ

Що оцінюємо протягом семестру?

- Вчасність та повноту виконання поставленого завдання
- Вмотивованість та уміння працювати в команді
- Уміння неординарно вирішувати поставлену проблему, брати ініціативу
- Уміння пропонувати свої ідеї та шляхи їх реалізації для загального обговорення
- Повнота засвоєння та глибина розуміння інструментарію (бібліотек, фреймворків і т.д.), що використовуються у проекті
- Уміння працювати із інформаційними-джерелами

Що оцінюємо протягом екзамену (демо)?

- Уміння представити проект (якість презентації, доповіді, повнота відповідей питання на поставлені запитання)
- Уміння аналізувати поставлене технічне завдання та планувати свою роботу
- Орієнтування у використаних технологіях, фреймворках, бібліотеках. Знати, які є аналоги
- Розбиратися в архітектурних рішеннях. Сильні/слабкі сторони використаної архітектури. Уміти обгрунтувати свій вибір
- Бачення вектору розвитку проекту та вміння оцінити його місце на ринку IT-послуг

ВІДКРИТІ ДЛЯ ЗАУВАЖЕНЬ ТА ПРОПОЗИЦІЙ

:-)

ДЯКУЄМО