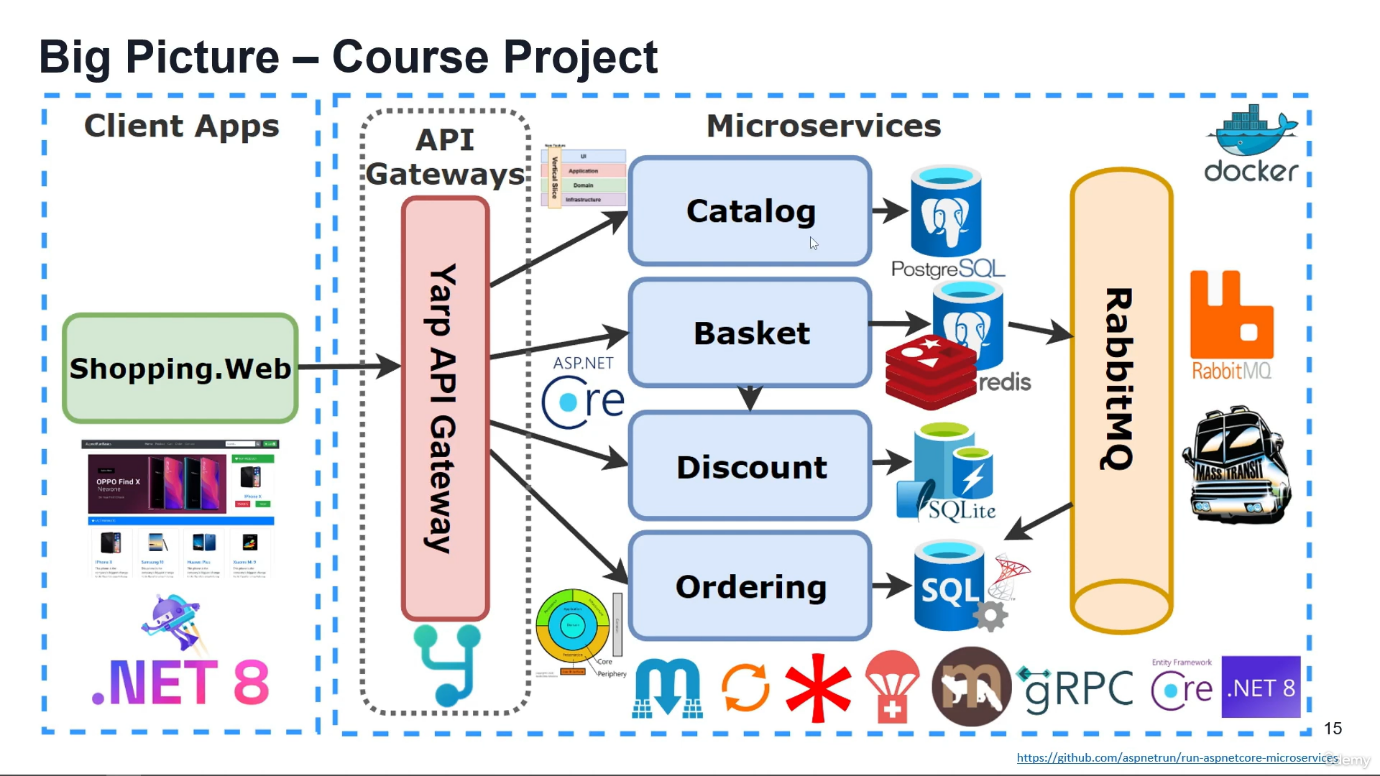
You will learn how to build**Microservices on .Net**platforms which used **Asp.Net Web API, Docker, RabbitMQ, MassTransit, Grpc, Yarp API Gateway, PostgreSQL, Redis, SQLite, SqlServer, Marten, Entity Framework Core, CQRS, MediatR, DDD, Vertical**and **Clean Architecture**implementation using **latest codes**and **best practices**of **.NET 8**on cloud-native environments.



You will develop **e-commerce** modules over **Product, Basket, Discount** and **Ordering** microservices with **NoSQL (PostgreSQL DocumentDB, Redis)** and **Relational databases (SQLite, Sql Server)** with communicating over **RabbitMQ Event Driven Communication** and using **Yarp API Gateway**. You can find **Microservices Architecture and Step by Step Implementation on .NET**which **step by step developing** this course with extensive explanations and details.

Along with this you’ll develop following microservices and items:

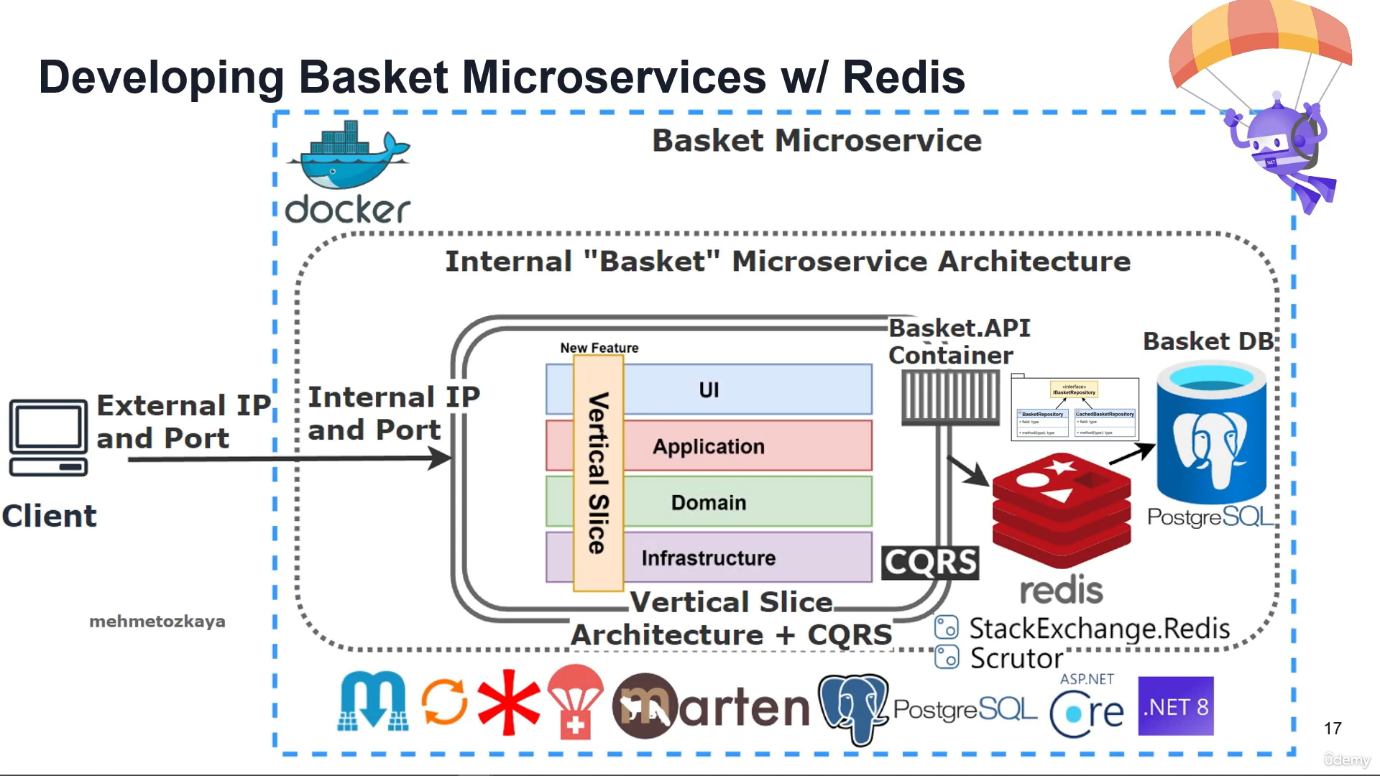
**Catalog microservice which includes;**

**A diagram of a microservice

Description automatically generated**

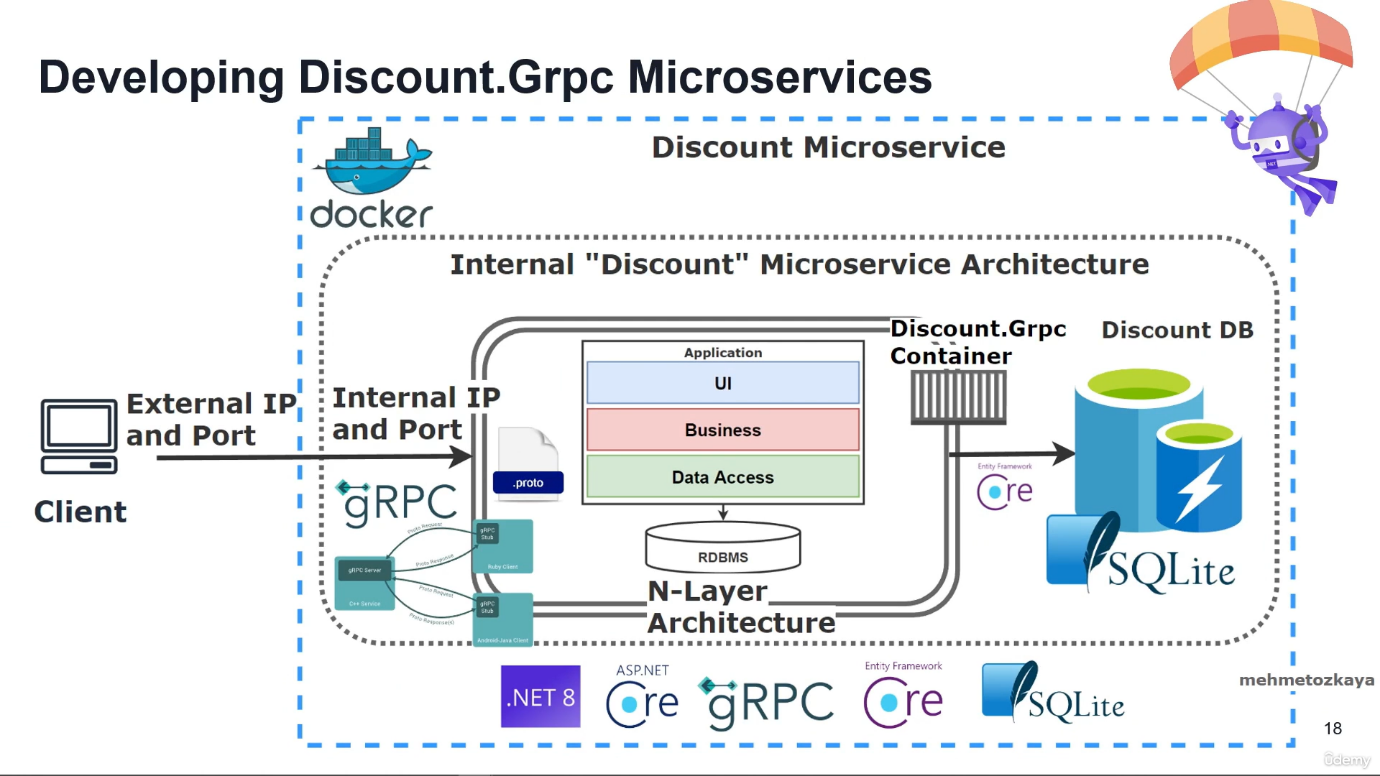
* ASP.NET Core Minimal APIs and latest features of **.NET 8**and **C# 12**
* **Vertical Slice Architecture** implementation with Feature folders
* **CQRS**implementation using **MediatR**library
* **CQRS**Validation Pipeline Behaviours with **MediatR**and **FluentValidation**
* **Marten**library for **.NET Transactional Document DB** on **PostgreSQL**
* **Carter**library for Minimal API endpoint definition
* Cross-cutting concerns **Logging**, global **Exception Handling**and **Health Checks**
* **Dockerfile**and **docker-compose** file for running Multi-container in Docker environment

**Basket microservice which includes;**



* **ASP.NET 8**Web API application, Following REST API principles, CRUD operations
* **Redis**as a **Distributed Cache**over basketdb
* Implements **Proxy**, **Decorator**and **Cache-aside**Design Patterns
* **Consume**Discount **gRPC Service** for inter-service sync communication to calculate product final price
* **Publish**BasketCheckout Queue with using **MassTransit**and **RabbitMQ**

**Discount microservice which includes;**



* ASP.NET **gRPC Server** application
* Build a Highly Performant **inter-service gRPC Communication** with Basket Microservice
* Exposing **gRPC Services**with creating **Protobuf messages**
* **Entity Framework Core ORM**- **SQLite** Data Provider and Migrations
* **SQLite database**connection and containerization

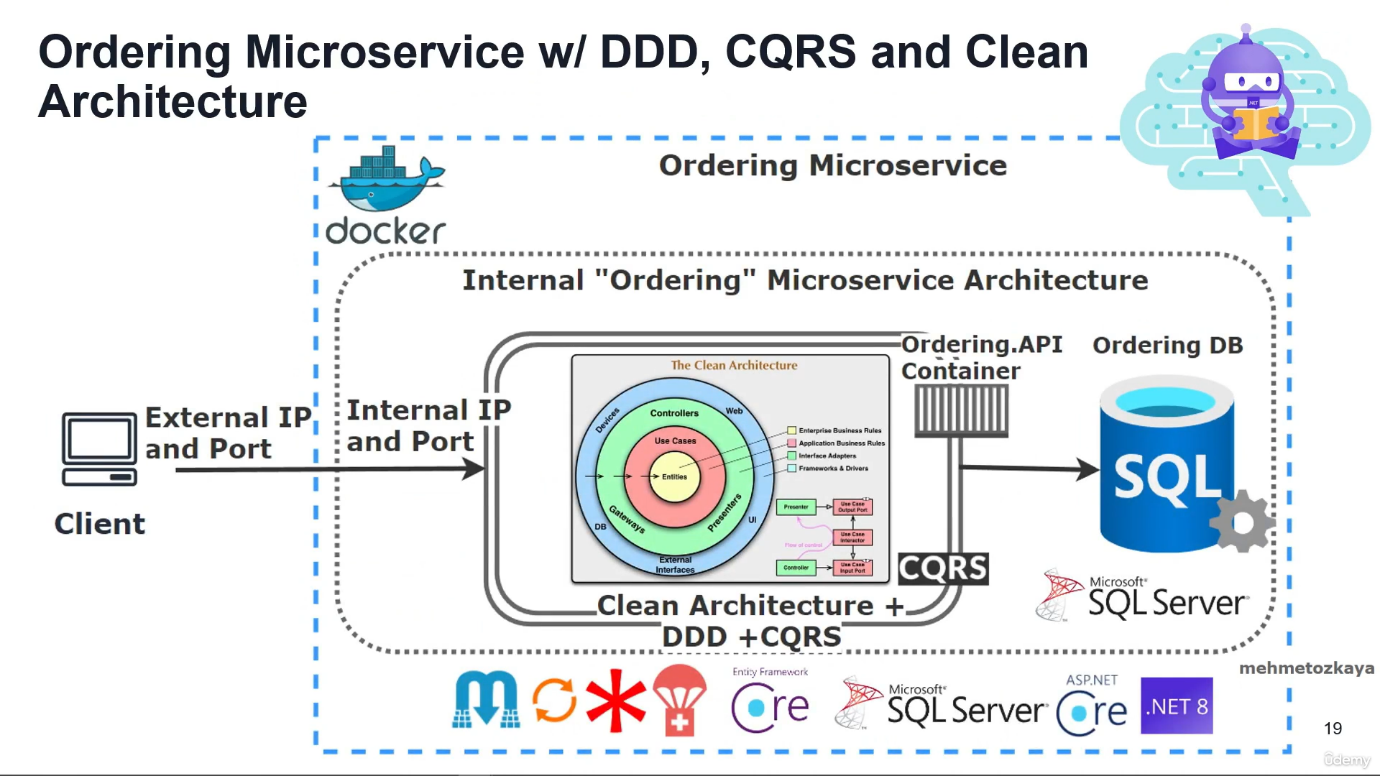
**Microservices Communication**

**A diagram of a business

Description automatically generated**

* Sync inter-service **gRPC Communication**
* Async Microservices Communication with **RabbitMQ Message-Broker Service**
* Using **RabbitMQ Publish/Subscribe Topic** Exchange Model
* Using **MassTransit** for abstraction over RabbitMQ Message-Broker system
* Publishing **BasketCheckout event**queue from Basket microservices and Subscribing this event from Ordering microservices
* Create **RabbitMQ EventBus.Messages library** and add references Microservices

**Ordering Microservice**



* Implementing **DDD, CQRS, and Clean Architecture** with using Best Practices
* Developing **CQRS with using MediatR, FluentValidation and Mapster packages**
* Use **Domain Events**& **Integration Events**
* **Entity Framework** Core**Code-First Approach**, **Migrations**, **DDD Entity Configurations**
* Consuming **RabbitMQ** BasketCheckout event queue with using **MassTransit-RabbitMQ** Configuration
* **SqlServer database** connection and containerization
* Using **Entity Framework Core ORM** and auto migrate to SqlServer when application startup

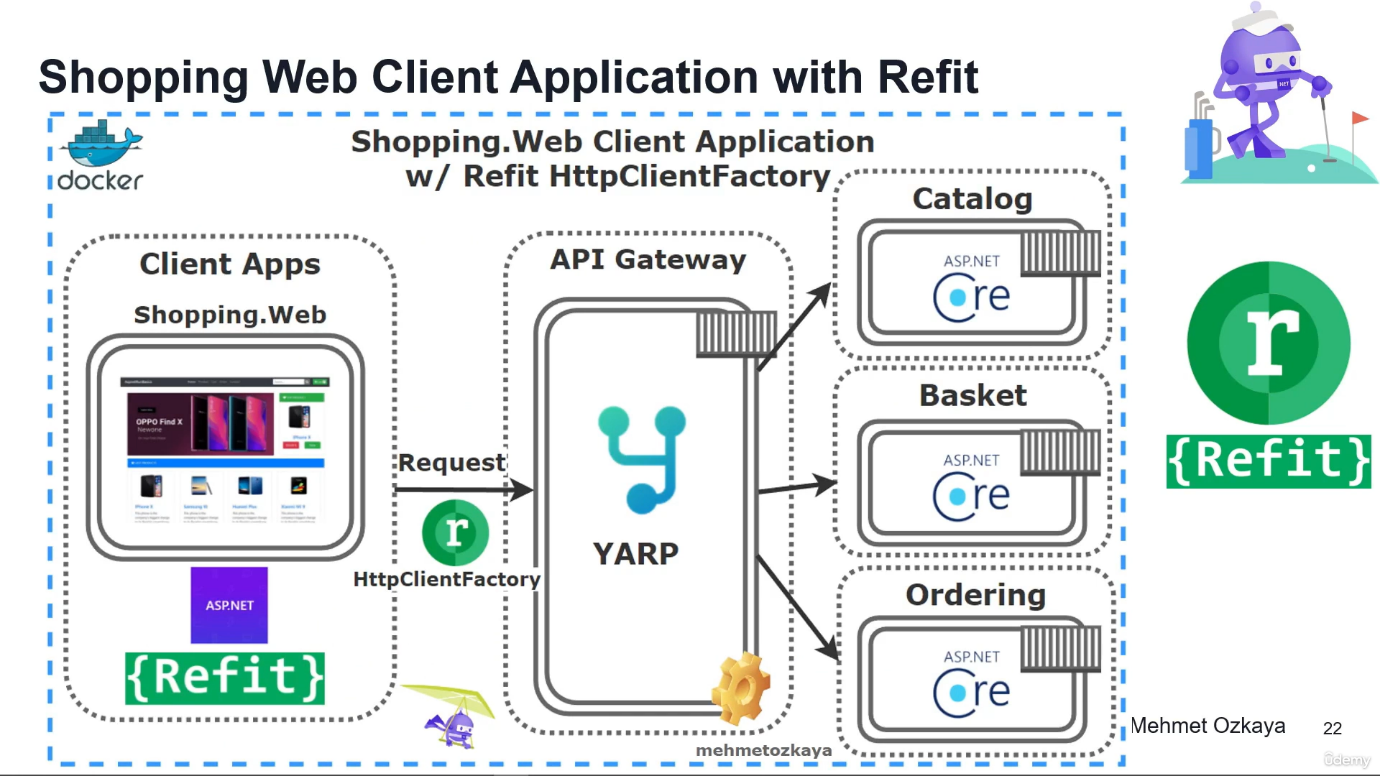
**Yarp API Gateway Microservice**

A diagram of a network

Description automatically generated

* Implement **API Gateways** with **Yarp Reverse Proxy** applying**Gateway Routing Pattern**
* **Yarp Reverse Proxy Configuration**; Route, Cluster, Path, Transform, Destinations
* **Rate Limiting** with **FixedWindowLimiter**on Yarp Reverse Proxy Configuration
* Sample microservices/containers to reroute through the API Gateways

**WebUI ShoppingApp Microservice**



* ASP.NET Core **Web Application**with**Bootstrap 4**and **Razor**template
* **Consume**YarpApiGateway APIs using **Refit Library** with **Generated HttpClientFactory**
* **ASPNET Core Razor Tools** — View Components, partial Views, Tag Helpers, Model Bindings and Validations, Razor Sections etc.

**Docker Compose establishment with all microservices on docker;**

* **Containerization**of microservices
* **Orchestrating**of microservices and backing services (databases, distributed caches, message brokers..)
* Override Environment variables

On top of all these, you'll learn **how to write quality code,**not just how to build microservices. In this course you will see the demonstrating a **layered application architecture** with **DDD**best practices. Implements NLayer **Hexagonal architecture** (Core, Application, Infrastructure and Presentation Layers) and **Domain Driven Design** (Entities, Repositories, Domain/Application Services, DTO's...) and aimed to be a **Clean Architecture**, with applying **SOLID principles** in order to use for a project template. Also implements **best practices** like **loosely-coupled, dependency-inverted** architecture and using **design patterns** such as **Dependency Injection**, logging, validation, exception handling and so on.