سیلابس دوره مایکروسرویس ، داکر و DDD و CQRS

معرفی:

در این سند به معرفی دقیق دوره پرداخته می شود. ابتدا عناوبن مورد بحث در دوره را بررسی می کنیم و بعد از آن جزئیات و اهداف هر جلسه بیان می شود. قابل ذکر است که این دوره در ۱۶ جلسه تدریس شده است و زمان هر جلسه دوساعت و نیم تا ۳ ساعت می باشد.

عناوین مورد بحث در دوره:

۱- داکر و اجرا مایکروسرویس ها برروی داکر و توسعه انواع ارتباطات بین مایکروسرویس‌ها:

این دوره را ابتدا با بیان مفاهیم Docker به صورت کاملا کاربردی شروع می‌کنیم و با ذکر چندین مثال از داکر به صورت مستقل، نحوه استفاده از آن به صورت شفاف آموزش داده می شود. در مرحله بعد با استفاده از DockerFile نحوه اجرای یک پروژه دات‌نت در Docker را شرح می دهیم و با آشنایی با Docker Compose به بیان مفهوم orchestration ، نحوه اجری چندین پروژه دات نت و همچنین وابستگی ها و ابزارهای مختلف در کانتینرهای مختلف به صورت هماهنگ شرح می دهیم. همچنین در داکر با بیان مفهوم state به نحوه نگهداری state ها در کانتینرهای می پردازیم.

در مرحله بعد با بیان اهمیت مایکروسرویس‌ها به توسعه و پیاده سازی دو سرویس ساده با پایگاه داده مختلف و غیر همجنس می‌پردازیم و ضمن پیاده‌سازی این دو مایکروسرویس به پیکربندی اولیه این دو مایکروسرویس که شامل راه اندازی آنها و پایگاه دادههای شان در docker و همچنین استفاده از serilog جهت لاگ‌گذاری و ورود اطلاعات اولیه آنها و مایگرت کردن آنها در پایگاه داده در محیط های cloud که با حالت عادی تفاوت هایی دارد و باید با توجه به نوع cloud (داکر یا مثلا سرویسkubernetes) سیاست های مختلف را اعمال کرد.

با توجه به اینکه اجرا در محیط های Cloud در مرحله اولیه شناخت سرویس ها و وابستگی‌های آنها از همدیگر ممکن است با چالش روبرو شوند، از همان مرحله اولیه توسعه سرویس‌ها با مفاهیمResilient Applications آشنا می شویم و با بهره گرفتن از ابزارهای Polly، الگوهای مختلفی سعی مجدد (Retry) را از همان ابتدا بکار می گیریم. قابل ذکر است که همانطور که در مراحل بعدی بیان می شود در بخش مجزایی این الگوها تشریح و استفاده می کنیم.

بعد از اتمام مراحل اجرا و پیکربندی اولیه دو مایکروسرویس ، با بیان یک مسئله چالشی مربوط به ارتباط و وابستگی بین این دو مایکروسرویس با هم، به پیاده‌سازی انواع روش‌های همزمان (Synchronous Communication) و غیرهمزمان (Asynchronous Communication) جهت ارتباط بین دو مایکروسرویس با هم می پردازیم و ابتدا با استفاده از روش RPC

دو سرویس را با هم ارتباط می دهیم و بعد از آن ضمن بیان مشکلات و محدودیت‌های RPC جهت ارتباط بین مایکروسرویس–ها، به استفاده از روش‌های تبادل پیام و Message Routing مراجعه می کنیم که کاملا Asynchronous هستند.

اولین روش Message Routing با استفاده از بیان مفهوم A Broker-Less System مانند ZerqMq به پیاده سازی ارتباط این دو مایکروسرویس می پردازیم و با استفاده از الگوی تولیدکننده و مصرف کننده در ZerqMq این ارتباط را توسعه می دهیم و به مزایا و معایب این روش پرداخته می شود و موقعیت های کاربردی و استفاده از آن را تشریح می کنیم.

در مرحله بعد به الگوهای Brokered روی می آوریم، و ضمن معرفی RabbitMQ و نحوه نصب و راه اندازی آن در Docker ، و همچنین معرفی آن به Docker Compose، به چگونگی ارتباط با RabbitMQ از مایکروسرویس ها، به نحوه تبادل پیام و الگوی تولید کننده و مصرف کننده در این نوع از ارتباطات می پردازیم...

قابل ذکر است که تمامی روش‌های فوق را به کارگیری سطحی از انتزاع به گونه ای پیاده سازی می کنیم که توسعه در انتخاب هر یک از روش‌های جهت ارتباطات مختلف آزاد باشد و هر کدام از آنها در جای مناسب استفاده کند.

همچنین در این قسمت مفاهیم مدل های مختلف مصرف پیام شامل Competing Consumers که شامل یک مصرف کننده است و Independent Consumers که شامل چندین مصرف کننده است را ذکر می کنیم و به پیاده سازی آنها می پردازیم.

در مرحله آخر در این قسمت بحث Delivery Guarantees نیز رجوع می کنیم و نکاتی که برای این مهم در ارسال پیام‌ها باید مدنظر قرار دهیم را شرح می دهیم.

۲ . DDDو CQRS

در این قسمت ابتدا با استفاده از الگوهای تجزیه (Business capability pattern و Sub-domain pattern)، نحوه تجزیه Domain را به چندین SubDomain تشریح می کنیم. سپس با بیان دقیق مفهوم Bounded Context که یک مرز ملموس کاربردپذیری از SubDomain است و به مجموعه کدهایی اطلاق می شود که business processes را در یک SubDomain پیاده سازی می‌کنند برای هر کدام از آنها یک مایکروسرویس در نظر میگیریم. در کنار این مایکروسرویس ها که جنبه عملیاتی دارند با بیان مفاهیم Technical Capabilities ، میکروسرویس های بیشتری را کشف می کنیم.

در مرحله بعد به کشف کلاس‌ها در هر Bounded Context می پردازیم و با استفاده از مفاهیم Aggregatesو Aggregate Roots، گروه‌های مختلف کلاس (Value Object،Entity و aggregate) را شناسایی می کنیم و نمایش می دهیم که چگونه یک Aggregate Roots ، باعث سازگاری اطلاعات در یک Bounded Context می‌شود.

در این قسمت مفاهیم دقیق Domain Services و Application Services را به صورت کاملا کاربردی بکار می گیریم و دقیق تفاوت این دو را در Domain Model شرح می دهیم و بیان می کنیم که این دو دقیقا چه تفاوت هایی دارند و دقیقا شرح می دهیم که چه زمانی باید کد را در سطح Domain و چه کدهایی باید در سطح Application توسعه داد.

با استفاده از روش Domain Driven Design ، هر Binding Context را به لایه های زیرتقسیم می کنیم:

- Domain
- Application
- API
- Infrastructure
- و...

لایه های Domain و Application که لایه‌های اصلی و عملیاتی هستند و لایه‌های API، Infrastructure جهت ارتباط با زیرساخت و منابع داده و دیگر ابزارهاست.

با استفاده از DomainEvent تمامی رویدادهای دامنه در لایه Domain Model شناسایی می کنیم و آنها را در هر Aggregate Rootتوسعه می دهیم و سپس را در زمان اعمال تغییرات به پایگاه داده، انتشارشان می دهیم که این انتشار ممکن است در برخی از حالات، منجر به انتشار Event به دیگر مایکروسرویس ها با استفاده از Broker شود.

با استفاده از الگوی CQRS و بکارگیری ابزار MediatorR ، روند فرمان (Command) و پرس و جوها (Query) را از همدیگر جدا می کنیم که باعث افزایش کارائی به دلیل قفل نشدن پایگاه داده می شود.

در قسمت Command ها یاد می گیریم که چگونه Application Service ها را به صورت جعبه سیاه طراحی کنیم، به گونه ای که کاملا مستقل توسط Api یا رویدادهای مختلف فرخوانی شوند. در این حالت منطق عملیات به صورت کاملا مجزا پیاده‌سازی شده است.

در این قسمت با تفکیک پایگاه داده نوشتن و خواندن و بکارگیری مفهوم Eventual Consistency روش های مختلف همزمانی دو پایگاه داده، بررسی می کنیم و همچنین با بکارگیری مدل داده مجزا (Read Data Model و Write Data Model) برای هر یک از این پایگاه داده‌ها و بیان مفهوم Materialized view به راحتی می توانیم برروی هر یک از این پایگاه داده خصوصا برای عملیات های خواندن سیاست‌های مختلفی مانند Index گذاری تعریف کنیم. به عنوان مثال ممکن است Write DataBase به صورت JSON documents و ReadDataBase به صورت Relational باشد.

قابل ذکر است که برای پرس و جوها از ابزار Dapper استفاده می کنیم که به مراتب سرعت اجرای بالاتری را دارد.

با معرفی مفهوم مهم EventSource ، به عنوان یک اصل کاربردی در معماری مایکروسرویس به اهمیت ذخیره رویدادها در یک قسمت مجزا می پردازیم و نحوه ترکیب آن با الگوی CQRS را نیز شرح می دهیم و بیان می کنیم که چرا باعث سازگاری در پایگاه داده می شود و چرا باید در تراکنش ها اهمیت دارد.

۳. Api Gateway And Grpc

در این قسمت ابتدا تشریح می کنیم که چرا کلاینت ها نباید مستقیما با مایکروسرویس ها در ارتباط باشند و با تعریف Api Gateway مزایای مختلفی آن از جمله امنیت بهتر، مقیاس پذیری بلاتر و مدیریت راحتتر مایکروسرویس ها را به دست می آوریم. در مرحله بعد نوع ارتباط gRPC آشنا می شویم و ضمن بیان دقیق کاربرد آن، جهت ارتباط بین Gateway و مایکروسرویس ها از این نوع ارتباط بهره می بریم.

همچنین در این قسمت با بیان اینکه وجود ایجاد تنها یک نقطه دسترسی مشکلاتی را در مقیاس پذیری ایجاد می‌کند، از الگوی BFF جهت حل این مشکل استفاده می‌کنیم.

۴. Securing Microservices with IdentityServer4

با استفاده از IdentityServer4 ما می توانیم یک نقطه دسترسی واحد برای کنترل دسترسی و Authentication مهیا کنیم که تمامی مایکروسرویس ها در درجه اول توسط همین نقطه دسترسی و در مرحله بعد از طریق اعمال امنیت در Api Gateway، محافظت می شوند و با استفاده از مفاهیم پروتکل OAuth2 را برای آنها authorization را مهیا می کنیم.

اهداف جلسات:

هدف جلسه اول:

۱. **معرفی داکر و دلایل استفاده از آن:**
   - توضیح مفهوم داکر و تفاوت آن با مجازی‌سازی سنتی.
   - بررسی مزایای استفاده از داکر در توسعه و استقرار نرم‌افزار.
۲. **نصب و راه‌اندازی داکر:**
   - آموزش نصب Docker Desktop و تنظیمات اولیه.
   - معرفی دستورات پایه داکر مانند docker build، docker run، docker psو docker images.
۳. **ساخت و اجرای اولین کانتینر:**
   - ایجاد یک وب سرور ساده با استفاده از nginxو اجرای آن در یک کانتینر.
   - بررسی نحوه ساخت Dockerfileو استفاده از دستورات پایه داکر.
٤. **مدیریت کانتینرها و تصاویر:**
   - آموزش دستورات مدیریتی مانند docker rm، docker rmiو docker exec.
   - بررسی نحوه استفاده از docker logsو docker inspectبرای تحلیل کانتینرها.
٥. **کار با:Docker Hub**
   - آموزش ثبت و مدیریت تصاویر در.Docker Hub
   - استفاده از دستورات docker pushو docker pullبرای به‌اشتراک‌گذاری و دریافت تصاویر.
٦. **اجرای پایگاه داده SQL Server در داکر:**
   - آموزش اجرای SQL Server در یک کانتینر و اتصال به آن.
   - بررسی نحوه استفاده از متغیرهای محیطی برای پیکربندی.SQL Server
۷. **مدیریت داده‌ها و Volume در داکر:**
   - توضیح مفهوم Volume و نحوه استفاده از آن برای ذخیره‌سازی داده‌ها به‌صورت پایدار.

- بررسی سناریوهای استفاده از Volume برای پایگاه‌داده‌ها.

۸. **راه‌اندازی اولین سرویس در پروژه:**

- ایجاد ساختار پروژه eShop-devsharp و راه‌اندازی سرویس CatalogService.Api.
- پیکربندی اولیه پروژه و اضافه کردن سرویس‌های ضروری مانند Swagger.

۹. **آشنایی با تصاویر رسمی دات‌نت:**

- معرفی تصاویر رسمی دات‌نت برای محیط‌های توسعه و تولید.
- بررسی تفاوت‌های بین تصاویر mcr.microsoft.com/dotnet/sdk و mcr.microsoft.com/dotnet/aspnet.

اهداف جلسه دوم: پیکربندی داکر و مدیریت سرویس‌ها

۱. **سازمان‌دهی پروژه**
   - انتقال Solution به پوشه srcو تنظیمات.WebHostEnvironment
   - مدیریت **appsettings.json**و .**appsettings.Development.json**
۲. **آشنایی با Docker Compose و اجرای چندین سرویس**
   - ساخت و پیکربندی فایل docker-compose.ymlبرای هماهنگی بین سرویس‌ها.
   - تنظیم وابستگی‌ها (Depends_On) برای اجرای پایگاه داده و سرویس‌ها.
۳. **مدیریت متغیرهای محیطی و امنیت در داکر**
   - تعریف **env files.**و مدیریت تنظیمات محیطی برای هر سرویس.
   - استفاده از چندین فایل docker-composeبرای محیط‌های مختلف (توسعه، تولید).
٤. **اتصال به پایگاه داده در داکر**
   - اجرای **SQL Server**در داکر و مدیریت Volume برای پایداری داده‌ها.
   - تنظیمات docker-compose.override.ymlبرای ارتباط سرویس‌ها.
٥. **راه‌اندازی لاگ‌گذاری پیشرفته باSerilog**
   - پیاده‌سازی **Serilog**و **Logstash**برای مدیریت لاگ‌های سیستم.
   - انتقال لاگ‌گذاری به **BuildingBlocks**برای استفاده در تمامی میکروسرویس‌ها.
٦. **مدیریت پایگاه داده و مهاجرت داده‌ها**
   - تنظیم **EF Core**برای اتصال به پایگاه داده و اجرای .**Migrations**
   - پیاده‌سازی استراتژی‌های **Connection Resiliency**برای بهبود پایداری ارتباطات.
۷. **آشنایی با الگوی Retry وPolly**
   - پیاده‌سازی الگوی **Retry**برای مقابله با مشکلات شبکه‌ای در محیط ابری.
   - استفاده از **Polly**برای کنترل رفتار خطاهای پایگاه داده و سرویس‌ها.

اهداف جلسه سوم: الگوهای طراحی، معماری DDD و پیاده‌سازی لایه دامنه

۱. **بهبود استراتژی Retry و جلوگیری از بار اضافی روی سیستم**
   - استفاده از **Exponential Backoff**و **Jitter**در سیاست‌های.Retry
   - بررسی شرایطی که نباید از Retry در **Kubernetes**استفاده شود.

۲. **ایجاد Building Block برای مدیریت Migration پایگاه داده**

- تعریف **Common.Host** برای اجرای خودکار Migration در تمام سرویس‌ها.
- پیاده‌سازی متد **MigrateDatabase** با الگوی Retry برای افزایش پایداری.

۳. **بهبود مدیریت تنظیمات و خطاها**

- ایجاد **AddCustomOptions** برای مدیریت بهتر پیام‌های خطای API.
- تنظیم قوانین **Validation** برای جلوگیری از درخواست‌های نامعتبر.

٤. **مقدمه‌ای بر معماری ماکروسرویس‌ها و DDD**

- تفاوت **Core Microservices** و **Supporting Microservices**.
- بررسی سبک‌های مختلف طراحی میکروسرویس و نقش DDD در آن.

٥. **مفاهیم اصلی DDD و پیاده‌سازی لایه Domain**

- بررسی **Bounded Context** و **Aggregates**.
- تفکیک **Domain Layer**، **Application Layer** و **Infrastructure Layer**.
- طراحی کلاس **Entity** و تعریف **Domain Events** برای مدیریت تغییرات.

٦. **پیاده‌سازی منطق دامنه و سرویس‌های موجودیت**

- مدیریت موجودی کالا: افزایش و کاهش موجودی با در نظر گرفتن محدودیت‌ها.
- اعمال قوانین تخفیف و اعتبارسنجی قیمت محصول.
- طراحی **Exception Handling** برای مدیریت خطاهای دامنه.

۷. **استانداردسازی مدل داده و بهینه‌سازی معماری ماکروسرویس**

- استفاده از **SeedWork** برای کاهش کدهای تکراری در موجودیت‌ها.
- بررسی **Entity Invariants** و اطمینان از معتبر بودن داده‌ها در تمام سناریوها.

اهداف جلسه سوم: الگوهای طراحی، معماری DDD و پیاده‌سازی لایه دامنه

۱. بهبود استراتژی Retry و جلوگیری از بار اضافی روی سیستم
   - استفاده از Exponential Backoff و Jitter در سیاست‌های Retry.
   - بررسی شرایطی که نباید از Retry در Kubernetes استفاده شود.
۲. ایجاد Building Block برای مدیریت Migration پایگاه داده
   - تعریف Common.Host برای اجرای خودکار Migration در تمام سرویس‌ها.
   - پیاده‌سازی متد MigrateDatabase با الگوی Retry برای افزایش پایداری.
۳. بهبود مدیریت تنظیمات و خطاها
   - ایجاد AddCustomOptions برای مدیریت بهتر پیام‌های خطای API.
   - تنظیم قوانین Validation برای جلوگیری از درخواست‌های نامعتبر.
٤. مقدمه‌ای بر معماری ماکروسرویس‌ها و DDD
   - تفاوت Core Microservices و Supporting Microservices.
   - بررسی سبک‌های مختلف طراحی میکروسرویس و نقش DDD در آن.
٥. مفاهیم اصلی DDD و پیاده‌سازی لایه Domain
   - بررسی Bounded Context و Aggregates.
   - تفکیک Domain Layer، Application Layer و Infrastructure Layer.
   - طراحی کلاس Entity و تعریف Domain Events برای مدیریت تغییرات.
٦. پیاده‌سازی منطق دامنه و سرویس‌های موجودیت
   - مدیریت موجودی کالا: افزایش و کاهش موجودی با در نظر گرفتن محدودیت‌ها.
   - اعمال قوانین تخفیف و اعتبارسنجی قیمت محصول.
   - طراحی Exception Handling برای مدیریت خطاهای دامنه.
۷. استانداردسازی مدل داده و بهینه‌سازی معماری ماکروسرویس

- استفاده از SeedWorkبرای کاهش کدهای تکراری در موجودیت‌ها.
- بررسی Entity Invariantsو اطمینان از معتبر بودن داده‌ها در تمام سناریوها.

اهداف جلسه چهارم:

۱. **پیکربندی:Entity Type Configuration**
   - آموزش نحوه پیکربندی Entity Type Configuration در EF Core.
   - استفاده از PropertyAccessMode.Fieldو HasColumnNameبرای تنظیمات دقیق‌تر.

۲. **اعمال قوانین کسب‌وکار:(Business Rules)**
   - اعمال قوانین مربوط به موجودی انبار و قیمت‌گذاری محصولات.
   - بررسی و اعتبارسنجی قوانین مانند maxStockThresholdو availableStock.

۳. **تولید Domain Event برای تغییرات قیمت:**
   - آموزش ایجاد DomainEventبرای تغییرات قیمت محصولات.
   - توضیح اهمیت اطلاع‌رسانی در صورت تغییر قیمت.

٤. **گسترش:Domain Model**
   - افزودن نیازمندی‌های جدید به Domain مانند مدیریت موجودی، تخفیف‌ها و قوانین مربوط به قیمت و نام محصولات.
   - تغییرات در ER Diagram با اضافه شدن موجودیت‌های جدید مانند Supplier.

۵. **طراحی Aggregate و:Navigation Property**
   - توضیح نحوه طراحی Aggregate و جلوگیری از استفاده مستقیم از Aggregate Rootدر دیگر Aggregate Root.
   - استفاده از Navigation Propertyبرای ارتباط بین موجودیت‌ها.

٦. **پیاده‌سازی Repository و:Unit of Work**
   - ایجاد IRepositoryو IUnitOfWorkبرای مدیریت عملیات دیتابیس.
   - توضیح اهمیت استفاده از IUnitOfWorkبرای مدیریت تراکنش‌ها در یک درخواست.HTTP

۷. **پیاده‌سازی:Domain Events**
   - توضیح مفهوم Domain Eventsو تفاوت آن با Integration Events.
   - نحوه استفاده از Domain Eventsبرای اطلاع‌رسانی و اجرای عملیات جانبی.(Side Effects)
   - استفاده از کتابخانه MediatRبرای مدیریت Domain Events.

۸. **پیاده‌سازی:Event Handlers**
   - ایجاد Event Handlersبرای مدیریت رویدادهایی مانند تغییر قیمت محصول یا کمبود موجودی.

- توضیح نحوه استفاده از Event Handlersبرای اجرای عملیات جانبی مانند اطلاع‌رسانی به مدیران یا ثبت درخواست به تامین‌کنندگان.

۹. استفاده از الگوی:CQRS

- توضیح اولیه درباره الگوی CQRS و نحوه استفاده از آن در جلسات آینده.

۱۰. پیاده‌سازی سرویس‌های مرتبط با:Supplier

- افزودن سرویس‌های جدید برای مدیریت Supplierو ارتباط آن با موجودیت‌های دیگر.
- ایجاد SupplierEntityTypeConfigurationو SeedSupplierبرای مقداردهی اولیه.

اهداف جلسه پنجم:

۱. **معرفی الگوی:(CQRS (Command Query Responsibility Segregation**
   - توضیح مفهوم CQRS و تفکیک عملیات‌های خواندن (Query) و نوشتن.(Command)
   - بررسی مزایای استفاده از CQRS در معماری میکروسرویس‌ها.

۲. **پیاده‌سازی Command و:Command Handler**
   - آموزش نحوه ایجاد Command و Command Handler برای عملیات‌های نوشتن.
   - بررسی ساختار Command و نحوه مدیریت آن در لایه‌های مختلف.

۳. **تفکیک عملیات‌های خواندن و نوشتن:**
   - توضیح نحوه تفکیک عملیات‌های خواندن و نوشتن در یک پایگاه داده.
   - بررسی سناریوهای پیشرفته‌تر مانند استفاده از دو پایگاه داده مجزا برای خواندن و نوشتن.

٤. **استفاده از Domain Events در:CQRS**
   - آموزش نحوه استفاده از Domain Events برای انتشار تغییرات در سیستم.
   - بررسی نحوه مدیریت تراکنش‌ها و Side Effects با استفاده از.Domain Events

۵. **پیاده‌سازی Query در:CQRS**
   - آموزش نحوه پیاده‌سازی عملیات‌های خواندن (Query) به‌صورت مستقل از مدل.DDD
   - استفاده از ابزارهایی مانند Dapper برای اجرایQuery ها.

٦. **مدیریت تراکنش‌ها با استفاده از:Pipeline Behavior**
   - آموزش نحوه مدیریت تراکنش‌ها در EF Core با استفاده از.Pipeline Behavior
   - بررسی نحوه استفاده از IDbContextTransactionبرای مدیریت تراکنش‌ها.

۷. **پیاده‌سازی:Logging Behavior**
   - آموزش نحوه ایجاد یک Pipeline Behavior برای لاگ‌گیری از عملیات‌های مختلف.
   - بررسی نحوه ثبت لاگ‌ها در طول اجرایCommand ها وQuery ها.

۸. **استفاده از Option Pattern برای پیکربندی:**
   - آموزش نحوه استفاده از IOptionsSnapshotبرای مدیریت تنظیمات پویا در برنامه.

اهداف جلسه ششم:

۱. **پیاده‌سازی Pipeline Behavior در MediatR:**

- آموزش نحوه ایجاد و استفاده از Pipeline Behavior برای مدیریت Concerns مانندLogging ، Validationو مدیریت تراکنش‌ها.
- بررسی نحوه استفاده از IPipelineBehaviorبرای افزودن رفتارهای سفارشی به Pipeline.

۲. **مدیریت تراکنش‌ها با استفاده از Pipeline Behavior:**

- آموزش نحوه مدیریت تراکنش‌ها در EF Core با استفاده از IDbContextTransaction.
- بررسی نحوه استفاده از BeginTransactionAsyncو CommitTransactionAsyncبرای مدیریت تراکنش‌ها در عملیات‌های مختلف.

۳. **پیاده‌سازی Logging Behavior:**

- آموزش نحوه ایجاد یک Pipeline Behavior برای لاگ‌گیری از عملیات‌های مختلف.
- بررسی نحوه ثبت لاگ‌ها در طول اجرایCommand ها وQuery ها.

٤. **مدیریت خطاها و اعتبارسنجی (Validation):**

- آموزش نحوه مدیریت خطاهای عمومی وDomain Exception ها با استفاده از فیلترهایASP.NET Core.
- بررسی تفاوت بین اعتبارسنجی در لایه Applicationو Domain.

٥. **استفاده از FluentValidation برای اعتبارسنجی:**

- آموزش نحوه استفاده از کتابخانه FluentValidationبرای اعتبارسنجیCommand ها وQuery ها.
- بررسی نحوه پیاده‌سازی اعتبارسنجی خودکار با استفاده از FluentValidation.AutoValidation.

٦. **پیاده‌سازی ValidatorBehavior:**

- آموزش نحوه ایجاد یک Pipeline Behavior برای اعتبارسنجی خودکارCommand ها وQuery ها.
- بررسی نحوه مدیریت خطاهای اعتبارسنجی و بازگرداندن پاسخ‌های مناسب به کلاینت.

۷. **تفکیک اعتبارسنجی در لایه‌های Application و Domain:**

- بررسی تفاوت بین اعتبارسنجی در لایه Applicationو Domain.
- آموزش نحوه اعمال قوانین کسب‌وکار (Business Rules) در لایه Domain و اعتبارسنجی‌های عمومی در لایه Application.

اهداف جلسه هفتم:

۱. **اعتبارسنجی (Validation) در برنامه‌های کاربردی:**

- آشنایی با مفهوم اعتبارسنجی و نقش آن در برنامه‌های کاربردی.
- استفاده از کتابخانه‌های مانند FluentValidationبرای اعتبارسنجی داده‌ها.
- نوشتن کلاس‌های Validator برای دستورات (Commands) و اعمال قوانین اعتبارسنجی.
- اعتبارسنجی خودکار (Auto Validation) و نحوه فعال‌سازی آن در پروژه.

۲. **اعتبارسنجی با استفاده از:Pipeline**

- ایجاد یک Pipeline به نام ValidatorBehaviorبرای اعمال اعتبارسنجی به صورت خودکار در هر درخواست.
- مدیریت خطاهای اعتبارسنجی و بازگرداندن پاسخ‌های مناسب به کلاینت.

۳. **اضافه کردن میکروسرویس:Discount**

- آشنایی با نحوه اضافه کردن یک میکروسرویس جدید به پروژه.
- پیکربندی و راه‌اندازی پایگاه داده Postgresبرای میکروسرویس.Discount
- استفاده از PgAdminبرای مدیریت پایگاه داده.

٤. **ارتباط بین میکروسرویس‌ها:**

- بررسی روش‌های ارتباط بین میکروسرویس‌ها (همزمان و غیرهمزمان).
- آشنایی با مفهوم (Remote Procedure Call) **RPC**و استفاده از HttpClientبرای ارتباط همزمان بین میکروسرویس‌ها.
- استفاده از IHttpClientFactoryبرای مدیریت بهتر درخواست‌های HTTP و افزایش قابلیت اطمینان.
- آشنایی با روش‌های غیرهمزمان مانند ارسال پیام‌های مبتنی بر رویداد (Event-Driven) و استفاده از پروتکل‌هایی مانند gRPCو .RabbitMQ

٥. **پیاده‌سازی ارتباط همزمان و غیرهمزمان:**

- پیاده‌سازی ارتباط همزمان با استفاده از HttpClientو .RPC
- پیاده‌سازی ارتباط غیرهمزمان با استفاده از پیام‌های مبتنی بر رویداد و پروتکل‌های مانند .RabbitMQ

اهداف جلسه هشتم:

۱. **ارتباط بین میکروسرویس‌ها با استفاده از روش‌های همزمان و غیرهمزمان:**
   - بررسی سناریوی ارتباط بین میکروسرویس‌های **Catalog**و **Discount**.
   - استفاده از روش‌های **همزمان** (Synchronous)مانند **RPC**و **HTTP**برای واکشی اطلاعات تخفیف.
   - استفاده از روش‌های **غیرهمزمان** (Asynchronous)مانند پیام‌های مبتنی بر رویداد (Event-Driven) برای انتشار تغییرات تخفیف.

۲. **پیاده‌سازی ارتباط همزمان با استفاده از HttpClientو RPC:**
   - استفاده از HttpClientبرای ارتباط همزمان با میکروسرویس **Discount**.
   - معرفی IHttpClientFactoryبرای مدیریت بهتر درخواست‌های HTTP و افزایش قابلیت اطمینان.
   - پیاده‌سازی **RPC** (Remote Procedure Call)با استفاده از HttpClient.

۳. **پیاده‌سازی ارتباط همزمان با استفاده از gRPC:**
   - معرفی gRPCبه عنوان یک فریم‌ورک مدرن و پرکاربرد برای ارتباطات همزمان.
   - مزایای استفاده از gRPCمانند عملکرد بالا، کاهش استفاده از شبکه و پشتیبانی از چندین زبان برنامه‌نویسی.
   - پیاده‌سازی gRPCدر میکروسرویس‌های **Catalog**و **Discount**.

٤. **استفاده از gRPC Interceptorsبرای مدیریت خطاها و لاگ‌گیری:**
   - معرفی Interceptorsدر gRPCبرای مدیریت خطاها و لاگ‌گیری.
   - پیاده‌سازی Client Interceptorsو Server Interceptorsبرای مدیریت خطاها و بهبود فرآیند ارتباطات.

٥. **مقایسه Middlewareو Interceptorsدر gRPC:**
   - بررسی تفاوت‌های بین Middlewareو Interceptorsدر gRPC.
   - مزایا و معایب هر یک از این روش‌ها در مدیریت درخواست‌ها و پاسخ‌ها.

٦. **آشنایی با سیستم‌های Broker-lessمانند ZeroMQو NetMQ:**
   - معرفی سیستم‌های **Broker-less**و مزایای آن‌ها در ارتباطات غیرهمزمان.
   - آشنایی با **NetMQ**به عنوان یک پیاده‌سازی از **ZeroMQ**در دات نت.

اهداف جلسه نهم:

۱. **ادامه بحث gRPC Interceptorsو مدیریت خطاها:**
   - بررسی Client Interceptorsو Server Interceptorsدر gRPC.
   - مدیریت خطاها در gRPCبا استفاده از Interceptors.
   - پیاده‌سازی Interceptorsبرای لاگ‌گیری و مدیریت خطاها در سمت کلاینت و سرور.
۲. **مقایسه Middlewareو Interceptorsدر gRPC:**
   - بررسی تفاوت‌های بین Middlewareو Interceptorsدر gRPC.
   - مزایا و معایب هر یک از این روش‌ها در مدیریت درخواست‌ها و پاسخ‌ها.
۳. **ارتباطات غیرهمزمان (Asynchronous) و پیام‌های مبتنی بر رویداد:(Event-Driven)**
   - معرفی ارتباطات غیرهمزمان و مزایای آن‌ها در معماری میکروسرویس‌ها.
   - بررسی سیستم‌های **Broker-less**مانند **ZeroMQ**و **NetMQ**.
   - پیاده‌سازی ارتباطات غیرهمزمان با استفاده از **NetMQ**در **ASP.NET Core**.
٤. **معرفی و پیاده‌سازی Integration Events:**
   - ایجاد ساختار مشترک برای مدیریت Integration Eventsبین میکروسرویس‌ها.
   - استفاده از Integration Eventsبرای ارتباطات غیرهمزمان بین میکروسرویس‌ها.
٥. **اضافه کردن میکروسرویس Basket:**
   - معرفی میکروسرویس Basketو استفاده از پایگاه داده **Redis**.
   - پیاده‌سازی CQRSو DDDدر میکروسرویس Basket.
   - بررسی نحوه ذخیره‌سازی و بازیابی داده‌ها در **Redis**.
٦. **پیاده‌سازی gRPCدر میکروسرویس Basket:**
   - استفاده از gRPCبرای ارتباطات همزمان بین میکروسرویس‌ها.
   - مدیریت خطاها و برگشت کدهای gRPCدر میکروسرویس Basket.

اهداف جلسه دهم:

۱. **پیاده‌سازی Event Handler با لاگ‌گذاری:**
   - آموزش نحوه پیاده‌سازی DiscountCreatedEventHandlerبا استفاده از لاگ‌گذاری برای ردیابی رویدادها.

۲. **معرفی و پیاده‌سازی میکروسرویس:Basket**
   - بررسی ساختار میکروسرویس Basket و استفاده از اصول DDD و.CQRS
   - معرفی پایگاه داده Redis و مزایای استفاده از آن برای ذخیره‌سازی داده‌های سبد خرید.

۳. **استفاده از Docker Compose برای راه‌اندازی میکروسرویس:Basket**
   - آموزش نحوه تنظیم docker-composeو docker-compose.overrideبرای راه‌اندازی میکروسرویس Basketو.Redis

۴. **معرفی و استفاده از:Redis**
   - بررسی نحوه ذخیره‌سازی و بازیابی داده‌ها در.Redis
   - توضیح مزایای استفاده از ConnectionMultiplexerبرای مدیریت اتصالات به.Redis

۵. **پیاده‌سازی Repository برای:Redis**
   - آموزش نحوه پیاده‌سازی Repository برای دسترسی به داده‌های ذخیره‌شده در.Redis
   - بررسی نحوه خواندن کلیدها از Redis و مدیریت داده‌ها.

۶. **معرفی و استفاده از:gRPC**
   - توضیح نحوه استفاده از gRPC برای ارتباط بین میکروسرویس‌ها.
   - بررسی نحوه برگشت کدهای gRPC و مدیریت خطاها.

۷. **اضافه کردن رویداد :CatalogPriceChange**
   - آموزش نحوه اضافه کردن رویداد CatalogPriceChangedIntegrationEventو مدیریت آن در میکروسرویس‌ها.
   - بررسی نحوه استفاده از ZeroMQ برای انتشار رویدادها.

۸. **معرفی مفهوم Eventual Consistency و:CAP Theorem**
   - توضیح تفاوت بین Strong Consistency و.Eventual Consistency
   - بررسی CAP Theorem و انتخاب بینConsistency ، Availabilityو Partition Tolerance در معماری میکروسرویس‌ها.

۹. **معرفی و استفاده از:(RabitMQ) Message Broker**

- آموزش نحوه عملکرد RabbitMQ و مزایای استفاده از آن برای ارتباطات ناهمزمان.
- بررسی مفاهیم Channel و Exchange در RabbitMQ.

۱۰. استفاده از MassTransit برای مدیریت Event Bus:

- آموزش نحوه پیکربندی و استفاده از MassTransit برای مدیریت رویدادها و ارتباطات بین میکروسرویس‌ها.
- بررسی نحوه انتشار رویدادها با استفاده از IPublishEndpoint.

اهداف جلسه یازدهم:

۱. **تحلیل و پیاده‌سازی RabbitMQ برای ارتباطات غیرهمزمان:**
   - بررسی مفاهیم **Delivery Guarantees** در RabbitMQ مانند **At Most Once** و **At Least Once**.
   - پیاده‌سازی الگوی **Outbox Pattern** برای مدیریت پیام‌ها و جلوگیری از ناسازگاری در سیستم.
   - بررسی روش‌های مختلف برای حل مسئله ناسازگاری در سیستم‌های توزیع‌شده.

۲. **پیاده‌سازی Outbox Pattern در میکروسرویس‌ها:**
   - ایجاد جدول IntegrationEventLogEntry برای ذخیره‌سازی رویدادها و مدیریت وضعیت آن‌ها.
   - پیاده‌سازی Worker برای پردازش دوره‌ای پیام‌های Outbox و ارسال آن‌ها به RabbitMQ.
   - مدیریت خطاها و ارسال مجدد پیام‌ها در صورت شکست.

۳. **مدیریت خطاها در میکروسرویس‌ها:**
   - ایجاد یک سیستم مدیریت خطای یکپارچه برای میکروسرویس‌ها.
   - پیاده‌سازی CatalogApplicationException و CatalogErrorHandler برای مدیریت خطاها در لایه‌های مختلف.
   - بهبود مدیریت خطاها در gRPC و HTTP با استفاده از Interceptors و Middleware.

٤. **تفاوت بین Event و Command در ارتباطات غیرهمزمان:**
   - بررسی تفاوت‌های بین Event و Command در معماری میکروسرویس‌ها.
   - پیاده‌سازی Command با استفاده از Request/Response در RabbitMQ با کمک MassTransit.
   - استفاده از Saga و StateMachine برای مدیریت تراکنش‌های توزیع‌شده.

۵. **پیاده‌سازی Request/Response با استفاده از MassTransit:**
   - ایجاد Command و Response برای ارتباطات غیرهمزمان بین میکروسرویس‌ها.
   - پیاده‌سازی Request/Response در RabbitMQ با استفاده از MassTransit.
   - مدیریت خطاها و بازگشت پاسخ‌های مناسب به کلاینت.

اهداف جلسه دوازدهم:

۱. **مدیریت خطاها در میکروسرویس‌ها:**
   - بررسی و پیاده‌سازی سیستم مدیریت خطا در میکروسرویس‌های مختلف مانند Basket و Catalog.
   - استفاده از BasketIdentityService برای مدیریت هویت کاربران در میکروسرویس Basket.

۲. **تفاوت بین Event و Command در ارتباطات غیرهمزمان:**
   - بررسی تفاوت‌های بین Event و Command در معماری میکروسرویس‌ها.
   - پیاده‌سازی Command با استفاده از Request/Response در RabbitMQ با کمک MassTransit.

۳. **پیاده‌سازی Request/Response با استفاده از MassTransit:**
   - ایجاد Command و Response برای ارتباطات غیرهمزمان بین میکروسرویس‌ها.
   - پیاده‌سازی Request/Response در RabbitMQ با استفاده از MassTransit.
   - مدیریت خطاها و بازگشت پاسخ‌های مناسب به کلاینت.

٤. **معرفی و پیاده‌سازی میکروسرویس Ordering:**
   - بررسی سناریو و نیازمندی‌های میکروسرویس Ordering.
   - پیاده‌سازی عملیات ثبت سفارش و مدیریت فرآیند سفارشات.
   - استفاده از Domain Events و Integration Events برای مدیریت رویدادها در میکروسرویس Ordering.

٥. **پیاده‌سازی Value Objects در میکروسرویس Ordering:**
   - بررسی مفهوم Value Objects و تفاوت آن با Entities.
   - پیاده‌سازی Value Objects مانند Address در میکروسرویس Ordering.

٦. **پیاده‌سازی Checkout در میکروسرویس Basket:**
   - بررسی فرآیند Checkout و تبدیل اطلاعات سبد خرید به سفارش.
   - پیاده‌سازی Command و Handler برای مدیریت فرآیند Checkout.

اهداف جلسه سیزدهم:

۱. **معرفی و پیاده‌سازی الگوی Sagaبرای مدیریت تراکنش‌های توزیع‌شده:**
   - بررسی مفهوم Sagaو تفاوت آن با تراکنش‌های توزیع‌شده.
   - معرفی الگوهای مختلف Sagaمانند Choreographyو .Orchestration

۲. **پیاده‌سازی الگوی Orchestrationبا استفاده از :State Machine**
   - بررسی و پیاده‌سازی State Machineبرای مدیریت فرآیند سفارشات در میکروسرویس .Ordering
   - استفاده از ابزار Automatonymousبرای پیاده‌سازی State Machineدر دات نت.

۳. **پیاده‌سازی State Machineدر میکروسرویس :Ordering**
   - تعریف وضعیت‌ها(States) ، رویدادها (Events) و رفتارها (Behaviors) در .State Machine
   - پیاده‌سازی فعالیت‌ها (Activities) برای هر وضعیت در .State Machine

٤. **مدیریت خطاها و عملیات‌های جبرانی:(Compensating Transactions)**
   - بررسی نحوه مدیریت خطاها و اجرای عملیات‌های جبرانی در .State Machine
   - پیاده‌سازی عملیات‌های جبرانی برای بازگشت به وضعیت اولیه در صورت بروز خطا.

٥. **پیاده‌سازی ارتباط بین میکروسرویس‌ها با استفاده از :State Machine**
   - بررسی نحوه ارتباط بین میکروسرویس‌های ,Ordering, Catalogو Paymentبا استفاده از .State Machine
   - پیاده‌سازی Commandو Eventبرای مدیریت فرآیند پرداخت و بررسی موجودی کالاها.

٦. **پیاده‌سازی Background Serviceبرای مدیریت فرآیندهای خودکار:**
   - ایجاد Background Serviceبرای استخراج سفارش‌های با وضعیت Submittedو ارسال رویدادهای مربوطه به State Machine.

اهداف جلسه چهاردهم:

۱. **ادامه پیاده‌سازی State Machine در میکروسرویس Ordering:**
   - بررسی و پیاده‌سازی فعالیت‌های مربوط به هر وضعیت (State) در State Machine.
   - پیاده‌سازی عملیات‌های جبرانی (Compensating Operations) در صورت بروز خطا.

۲. **پیاده‌سازی State Machine برای مدیریت سفارشات:**
   - بررسی وضعیت‌های مختلف سفارش مانند Submitted, AwaitingValidation, StockConfirmed, StockRejected, Paid, و Completed.
   - پیاده‌سازی فعالیت‌های مربوط به هر وضعیت مانند OrderSubmittedActivity, StockRejectedActivity, StockConfirmedActivity, و PaidActivity.

۳. **پیاده‌سازی ارتباط با میکروسرویس‌های دیگر:**
   - ارسال Command به میکروسرویس Catalog برای بررسی موجودی کالاها.
   - ارسال Command به میکروسرویس Payment برای انجام عملیات پرداخت.

٤. **مدیریت خطاها و Idempotency در میکروسرویس‌ها:**
   - بررسی مفهوم Idempotency و اهمیت آن در سیستم‌های توزیع‌شده.
   - پیاده‌سازی Idempotency برای جلوگیری از اجرای چندباره عملیات‌های حساس مانند ثبت سفارش.

۵. **پیاده‌سازی Idempotency در میکروسرویس Ordering:**
   - ایجاد جدول Requests برای ذخیره‌سازی درخواست‌های تکراری.
   - پیاده‌سازی IdentifiedCommandHandler برای مدیریت درخواست‌های تکراری.

٦. **انتشار رویدادها به کلاینت:**
   - بررسی نحوه انتشار رویدادهای تغییر وضعیت سفارش به کلاینت‌ها با استفاده از WebSocket یا SignalR.

اهداف جلسه پانزدهم:

۱. **معرفی و پیاده‌سازی الگوی API Gateway:**
   - بررسی مفاهیم و مزایای استفاده از API Gatewayدر معماری میکروسرویس‌ها.
   - پیاده‌سازی API Gatewayبا استفاده از Envoyبه عنوان یک پروکسی سبک‌وزن و پرکاربرد.

۲. **پیاده‌سازی API Gatewayبا Envoy:**
   - تعریف فایل پیکربندی Envoyبرای مدیریت مسیرها و کلاس‌ترها.
   - تنظیم Docker-Composeبرای اجرای Envoyبه عنوان API Gateway.

۳. **استفاده از الگوی Gateway Aggregation Pattern:**
   - بررسی و پیاده‌سازی الگوی Gateway Aggregationبرای تجمیع درخواست‌ها از چندین میکروسرویس.
   - ایجاد یک Aggregatorبرای مدیریت درخواست‌های فرانت‌اند و تجمیع داده‌ها از چندین میکروسرویس.

٤. **مدیریت Swaggerدر API Gateway:**
   - تنظیم Swaggerبرای نمایشAPI های تمامی میکروسرویس‌ها در یک URLواحد.
   - مدیریت درخواست‌های فرانت‌اند که نیاز به واکشی اطلاعات از چندین میکروسرویس دارند.

٥. **پیاده‌سازی Service Discoveryو Service Registry:**
   - بررسی مفهوم Service Discoveryو اهمیت آن در معماری میکروسرویس‌ها.
   - معرفی ابزارهای مختلف برای پیاده‌سازی Service Discoveryمانند Consulو Eureka.

٦. **پیاده‌سازی Load Balancingدر Envoy:**
   - بررسی روش‌های مختلف Load Balancingدر Envoyو استفاده از روش Round Robin.

اهداف جلسه شانزدهم:

۱. **معرفی و پیاده‌سازی Identity Server 4:**
   - بررسی مفاهیم پایه‌ای Identity Server 4 و نقش آن در احراز هویت (Authentication) و مجوزدهی (Authorization).
   - توضیح مفاهیم OAuth 2.0 و OpenID Connect و نحوه استفاده از آنها در Identity Server 4.

۲. **پیاده‌سازی Identity Server 4 در معماری میکروسرویس‌ها:**
   - آموزش نحوه تنظیم و پیکربندی Identity Server 4 برای مدیریت کاربران، نقش‌ها و کلاینت‌ها.
   - بررسی نحوه ذخیره‌سازی اطلاعات کاربران و کلاینت‌ها در پایگاه داده‌های مختلف با استفاده از Entity Framework.

۳. **معرفی انواع Grant Type در OAuth 2.0:**
   - بررسی انواع Grant Type مانند Client Credentials, Resource Owner Password, و غیره.
   - آموزش نحوه استفاده از هر یک از Grant Type ها برای احراز هویت و صدور توکن.

٤. **پیاده‌سازی احراز هویت و مجوزدهی در میکروسرویس‌ها:**
   - آموزش نحوه امن‌سازی میکروسرویس‌ها مانند Basket API و Ordering API با استفاده از توکن‌های JWT.
   - بررسی نحوه استفاده از JwtBearer برای احراز هویت و مدیریت دسترسی‌ها.

٥. **مدیریت توکن‌ها و Refresh Token:**
   - توضیح مفهوم Refresh Token و نحوه استفاده از آن برای تمدید توکن‌های منقضی شده.
   - بررسی تنظیمات مربوط به طول عمر توکن‌ها و نحوه مدیریت آنها.

٦. **پیاده‌سازی Profile Service:**
   - آموزش نحوه اضافه کردن اطلاعات کاربری مانند نقش‌ها و Claim ها به توکن‌های JWT.
   - بررسی نحوه استفاده از IProfileService برای مدیریت اطلاعات کاربری.

۷. **امن‌سازی Gateway با استفاده از Identity Server 4:**
   - آموزش نحوه امن‌سازی Gateway با استفاده از توکن‌های JWT و مدیریت دسترسی‌ها.

۸. **استفاده از Docker Compose برای راه‌اندازی Identity Server:**
   - آموزش نحوه تنظیم docker-compose برای راه‌اندازی Identity Server و سایر میکروسرویس‌ها.