Micro Service

یک طراحی بیرونی است که به معنی شکستن یک برنامه بزرک به قطعات کوچک است.

فرض کنید که یک اتاق داریم پر از کارمند که با پارتیشن جدا کردیم مثلا فنی، حسابداری، ارتباطات، و در کنار هم کار می کنند و خروجی را به مدیر عامل می دهند.

اما با ماکروسرویس می توانیم هر بخش را جدا کرده به ساختمان های در محل های دیگر بفرستیم و در نهایت آن بخش ها کار خود را خروجی گرفته و به مدیر عامل بدهند.

**Soa**

معماری قدیمی که نام آن service oriented است که اپلیکیشن را به بخش های کوچکتری تبدیل می کند. در واقع میکروسرویس وام گرفته از آن است با این تفاوت که در میکروسرویس سرویس ها ما خورد تر میشود اما در soa در حد یک اپلیکیشن می تواند مستقل شوند.

**نکته :** در میکروسرویس می توان یک انبار را به سرویس های مستقلی تبدیل کرد به شرط این که مستقل بودن حفظ شود.

میکروسرویس در واقع External Architect است.

فرض کنید یک پروژه داریم با 3 تا سرویس a,b,c و هر کدام از این ها خود دارای یک معاری هستند.

یکی clean یکی union می تواند باشد.

**بین سرویس های می تواند ارتباط باشد به صورت 2 طرفه.**

**روش های ارتباط بین سرویس ها:**

GRPC

Google Remote Procedure Class

راه دوم Message Broker

به این صورت که کار ارتباط بین سرویس ها را انجام می دهد

چه کسی بگیرد.

ارسال و دریافت دیتا در یک صف و بررسی به ترتیب.

یک نرم افزار realTime است که صف را گوش می دهد و داده می گیرد و می فرستد.

RabbitMq – Kafka – redis – zmq – massTransiaent

**همه این ها از پروتکل Maqp استفاده می کنند.**

مانند یک تسمه عمل می کند داده را می برد و پاسخ را برمیگرداند.

**Standard**

Exchnage : مثل یک سری ادم هتسند که بسته ها رو تحویل آن ها می دهیم و می گوییم بزار تو فلان صف یعنی مشخص می کند که بسته ما در کدام صف قرار بگیرد چون می توانیم صف های مختلفی داشته باشیم و بگویم با چه قانونی در صف قرار بگیرد بسته.

برای ارتباط بین سرویس ها مثل سرویس user – information از روش prodeuce , consume یعنی ارائه دهنده و مصرف کننده که با rabbitMq حل است.

ابزار های مورد نیاز را می توانیم بر روی Docker بالا بیاوریم.

**Seq یک دیتا بیس مناسب برای ذخیره لاگ ها است.**

در این پروژه از Redis برای اطلاعات سبد خرید و elasticSearch نیز برای search کردن استفاده می کنیم.

**نکته: باید در میکروسرویس هر کدام از سرویس ها مستقل از هم باشند.**

**نکته »** باید توکن jwt در همه سرویس های تزریق شود به صورت common و در همه معتبر باشد.

**یک ApiGateWay می سازیم تا با استفاده از توکن از به سرویس های خود دسترسی داشته باشیم.**

**Cqrs** برای جدا کردن کوئری های select و insert می توان از database های مختلف به صورت cluster استفاده کرد به صورت always on avalibility group چند سرور داریم با چند sql که یک سرویس بین این ها run است به نام Failover یعنی یک سرور Primary داریم و مابقی secondery که به صورت sync و async می تواند باشد.

داده های نوشته شده روی جدول های primary بر روی secondery ها نیز نوشته می شود و اگر primary از بین برود یکی از secondery ها تبدیل به primary می شود و با ip به نام listener با هم در ارتباط هستند.

در Cqrs مثلا به یکی از secondery ها به صورت readOnly برای select زدن استفاده می کنیم.

MediatR برای پیاده سازی Cqrs کاربرد دارد.

ApiVersion هم برای ورژن بندی api ها کاربرد دارد

Second Lesson

Docker

یک سوپر وایزر است که در کنار ویندوز قرارمی گیرد و کار اجرای برنامه ها را انجام می دهد.

امکان نصب برنامه های مختلف رو می دهد مثل کی sql یک postgree می خواهیم بر روی داکر نصب می کنیم حتی برنامه های لینوکسی را هم می توانیم نصب کنیم چون هسته اصلی داکر لینوکس است

Image

یک بسته نرم افزار در داکر رو image می گوییم. یا مثالا یک سایت با کل dll هاش

Container

اجرای یک نرم افزار مثلا روی سرور را میگوییم کانتینر. می توانیم چندین image رو روی container بالا بیاوریم هر run را باید روی container انجام بدهیم.

Volumes

مکانی در سیستم عامل ما است که داده های نرم افزار ما روی آن اجرا می شود مثال فایل های mdf و ldf مربوط به sql server در آنجا قرار می گیرند.

Commands

دارای یک سری دستور است که با زدن docker لیست را میشود دید

Ps برای دیدن لیست برنامه های ران شده در چه پورتی و چخ ورژنی

Pull اگر بخواهیم یک نرم افزار را دانلود کنیم و روی داکر نصب کنیم از این دستور استفاده کنیم. دانلود شده ها را در images قرار می دهد و امکان اجرا می دهد.

Docker ps

Docker pull nginx

Docker images

برای مشاهده بسته های نرم افزاری نصب شده روی داکر

داکر یک فضای مجزا از سیستم عامل دارد با شبکه و سیستم عامل خاص خود.

**مثلا وقتی یک نرم افزار را دانلود کردی لازمه که روی داکر نصب شده :**

-d در بکگراند انجام بده

-p پورت اجرای داکر و سیستم عامل را می دهیم بهش

بعد کد نرم افزار که میخواهیم اجرا شود یا اسم نرم افزار را می گذاریم کد Digest

اگر پورت اشغال باشه پیام میده که عوضش کنی

بعد از اجرا در Container می توانیم ببینم اجرا شده است یا نه

امکان حذف از طریق کامند نیز وجود دارد.

**نکته: اول باید stop کرد بعد حذف.**

Docker run –d –p 8000:80 nginx

خود 80 و من 8000 می خواهم.

Rabbit Mq

از docker hub هم میتوانیم rabbit mq را دریافت کنیم یا در واقع command مناسب آن را پیدا کنیم برای نصب.

Redis

برای کش نیاز داریم از docker hub استفاده می کنیم.

موارد مهم جلسه دوم

لایه های معماری Clean

Api or Presentation

Domain لایه بیزنس ما است

Infra معمولا کوئری های ما اینجاست

Application در واقع لایه منطق ما است.

Ioc لایه که تزریق وابستگی خود را در آن قرار می دهیم.

قسمت سوم.

Cqrs به این معنی است که من میخواهم command و query خود را جدا کنم

Command دستوراتی که DataBase را تغییر میدهد می گویند. Delete update insert

Query دستوراتی که تغییری ندارد صرفا خواندن داده است مثل select

**فرض کنید:**

یک دیتابیس دارید که هم خواندن دیتا هم نوشتن دیتا رو روی ان دیتابیس انجام می شود و هم زمان دستور خواندن و نوشتن بر روی یک جدول دارید که سبب deadlock می شود برای رفع این خطا از 2 دیتابیس استفاده می کنیم که یکی صرفا برای نوشتن داده و دیگری خوادن داده و دیتابیس خواندن را با تاخیری اندک با دیتابیس نوشتن sync می کنیم به استفاده از روش availability group

**امکان پیاده سازی در یک cluster و 3 تا ماشین vm و تا 11 تا جاداره و 5 تا sync می شود.**

**ساختار cqrs**

Command فولدری ورودی های command در ان قرار میگیرد.

Query فولدری که ورودی های query در آن قرار میگیرد

**Handler فولدری که command ها یا query ها را پاسخ می دهیم.**

**نکته : از record برای داده های immutable یا داده ای که نمیخواهید تغییر دهید استفاده کنید.**

Annotation برای ولیدیشن های روی prop ها استفاده می شود مثل [Required]

**نکته :** کار لایه Application و فولدر Handler این است که از لایه infra داده می گیرد و داده می دهد و لایه infra نیز با لایه domain در ارتباط است.

نکته : UserHandler کلاسی است که با infra ارتباط دارد

**جلسه سوم**

**لایه domain**

در این لایه حق اضافه کردن هیچ پکیج ماژول و ... را نداریم و صرفا خود framework و زبان برنامه نویسی باید دخیل باشد. به هیچ لایه ای نباید متکی باشد به صورت کامل مستقل باید باشد و تمامی لایه ها می توانند به آن وابسته شود و قلب پروژه و مسئول اصلی بیزنس پروژه است.

Ddd

برای کم کردم پیچیدگی لایه domain است.

**Entity**

هر چیزی که **خصیصه** دارد **رفتار** دارد **ای دی یا مشخصه جداکننده** دارد.

Attribute – behavior – id

**نکته :** در بعضی موارد به جای int و انواع آن برای سطون id از نوع guid استفاده می شود که سبب کند شدن دیتابیس به دلیل جلوگیری از index گذاری صحیح داده ها می شود در صورتی که می توان از نوع داده های بزرگ تر هم مثل Int64 که معادل bigInt است استفاده کرد.

**موجودیت در سطح انتزاع abstract entity در سطح انتراع**

یک entity یا کلاسی که ویژگی های مشترکی از تمام object ها سرویس ها را نگه دارد. مثلا تصویر کارت ملی و شناسنامه ارتباطی به user به صورت مستقیم ندارد.

**نکته : لایه infra در واقع لایه persistence نیز هست یعنی می تواند با دیتابیس ارتباط بگیرد.**

**جلسه پنجم**

لایه application می تواند شامل:

Dto ها باشد

useCase ها یا خدماتی که به کاربران سیستم می دهیم و contract ها.

Api Version

با استفاده از پکیج api.versioning.http ممکن است.

روش های auth در microservice

Sso

Jwt از نوع تزریقی