معماری نرم افزار چیست؟

انتخاب یک ساخاتار کلی برای پیاده سازی یک پروژه نرم افزاری بر مبنای مجموعه ای از نیازهای کاربری و تجاری یکی سیستم نرم افزاری است تا هم بتوان کاربردهای مورد نظر را پیاده سازی کرد و هم بتوان کیفیت نرم افزار تولید آن و نگهداری آن را نیز بهینه کرد و سرعت بخشید.

ساختار باید بر اساس نیاز کسب و کار ما باشد و هر کسی که معماری را ببیند بتواند بفهمد هدف معماری چیست و مربوط به چه نوع پروژه ای است.

معماری باید کاربرد نرم فزار را نمایش دهد همانند نقشه یک ساختمان.

برای معماری نباید framework , tools مهم باشد بلکه useCase مهم است.

معماری باید ایپلیکیشن را خوب ساپورت کند و از ظاهرش مشخص باشد که مثلا برای اپلیکیشن حسابداری است.

معمار خوب کسی است که :

استفاده از ابزار ها و فریم ورک ها را به تاخیر بندازد با استفاده از abstraction , dependency inversion

هدف:

هدف اصلی حمایت از چرخه حیاط سیتم است معماری خوب باعث می شود درک، نگداری و استقرار سیستم آسان باشد. هدف نهایی به حداقل رساندن هزینه، افزایش طول عمرسیستم و به حداکثر رساندن بهره وری و برنامه نویسی است.

2 نوع از معروف ترین معماری ها: data centric - domain centric

Data centric

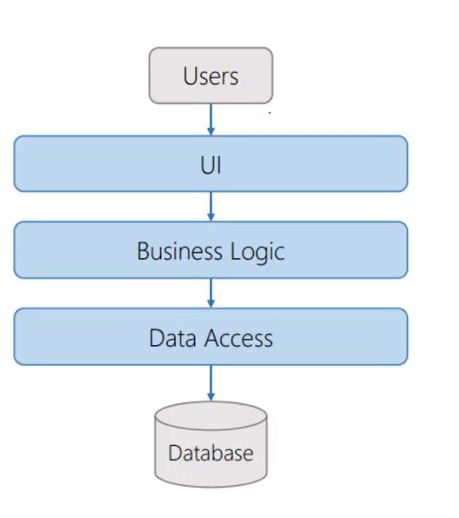
هسته اصلی نرم افزار DataBase است که خود اصول معماری نرم افزار را زیر سوال می برد اما برای حالت های که سریع می خواهیم اپلیکیشن را توسعه دهیم مناسب است اما خیلی سخت و کارایی پایین.

مزایا :

توسعه سریع

معایب :

دیتابیس مرکز نرم افزار است 0 انتخاب اشتباه مرکز ، testable نبودن ، پخش بودن قوانین کسب و کار

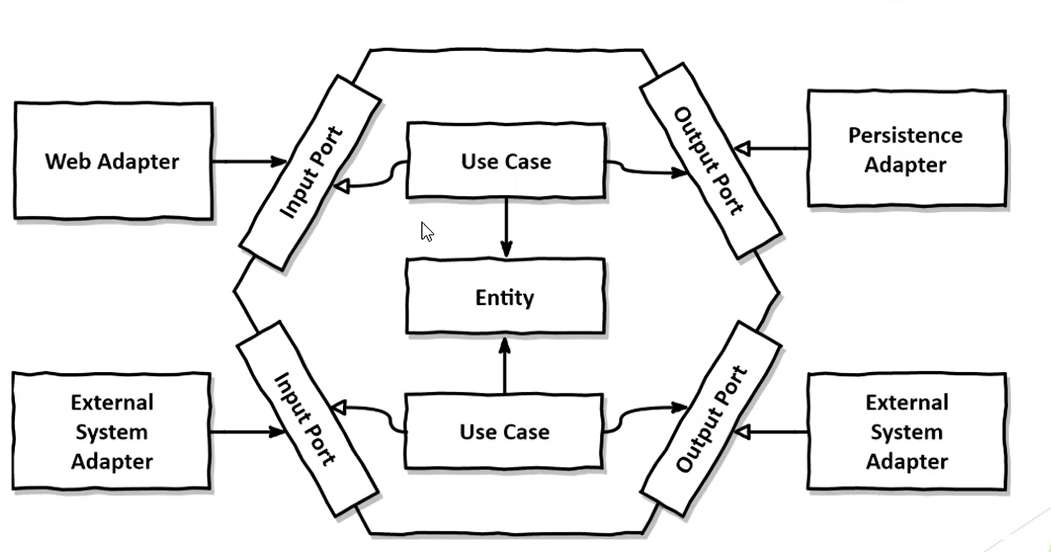


Domain centric

تمام قوانین کسب و کار و تمام موارد مربوط به کسب و کار در مرکز است و کاری با db و framework نداریم و تمامی پیش نیاز های نرم فزار را با توجه به امکانات زبان برنامه نویسی پیاده سازی کرده که سبب تست نویسی راحت تر و توسعه پذیری بیشتر می شود در ادامه مثال ها :

Hexagon

خیلی وقت است استفاده نمی شود



Onion

هسته مرکزی ما domain است و دارای یک applicationCore و قسمت های بیرونی داریم.

DomainModel

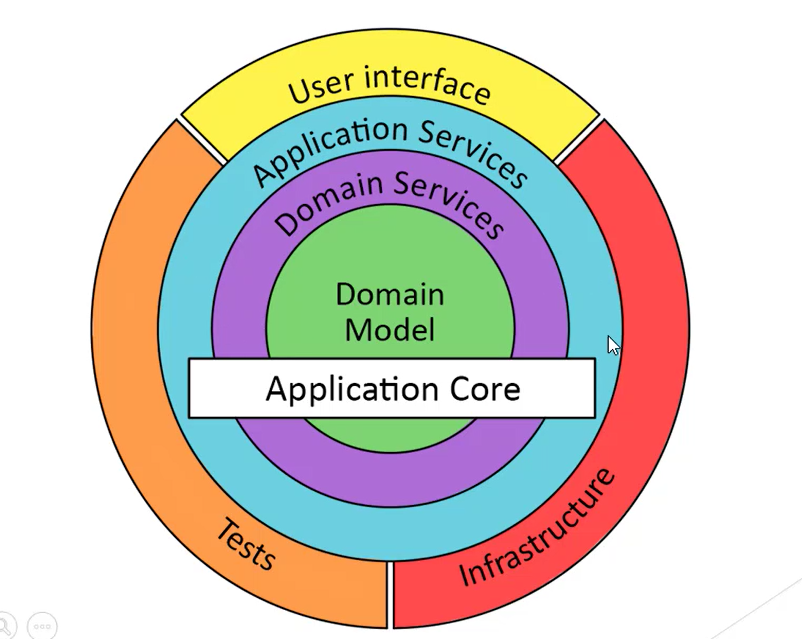
شامل entity ها و قوانین کسب و کار ما است

DomainService

سرویس های مربوط به entity ها و یک سری قوانین که در آن ها پیاده سازی شده

ApplicationService

useCase ها را شامل می شود مثلا در سیستم فروشگاهی یک useCase برای خرید مشتری داریم یک فانکشن که محصول را کم می کند به کاربر اضافه می کند و فرآیند را انجام می دهد.



Clean architecture

هدف این است که Core ما domain باشد نه database و از جزئیات دور باشیم تا زمانی که به آن نیاز داشته باشیم.

وابستگی ها به صورت لایه ای است هر لایه به قبلی خود وابسته است

Entites

در هسته مرکزی است و به هیچ چیزی وابسته نیست که قوانین کسب و کار است.

مثلا در ثبت نام کاربر شماره تکراری نیست این قانون کسب و کار است که در Entity پیاده سازی می شود.

UseCase

قوانین کسب و کار است

Interface adapter

مپ کردن دیتا برای ui و مپ کردن ui برای useCase ها را انجام می دهد کار input و output را انجام می دهد

لایه آخر

شامل framework ها و driver ها است

نکته: در معماری لزوما قرار نیست تمای لایه ها پیاده سازی شوند بلکه نسب به کار خود لایه ها را پیاده سازی می کنیم.

