تصمیمات مهم در توسعه نرم افزار با مقیاس زمینه ای که در ذهن ما است متفاوت است.یک مقیاس مهم که یک نرم افزار یا اپلیکیشن با آن سنجیده می شود، معماری نرم افزار یا اپلیکیشن است.اولین مسئله ای که باید در این آموزش به آن توجه کنیم، تعریف نرم افزار است، که در ادامه به آن می پردازیم.

**نرم افزار چیست؟**

اولین مشکلی که در این آموزش با آن درگیر هستیم اینست که تعریف واضح و دقیق از اپلیکیشن یا نرم افزار به طور تخصصی موجود نیست.به همین علت نرم افزار را از دیدگاه Martin Fowler بررسی خواهیم کرد.

اپلیکیشن ها یک ساختار اجتماعی هستند برای اقشار جامعه به عنوان مثال:

1. یک بدنه از کد که توسعه دهنده در یک واحد تنها می بیند.

2. یک گروه از عملکرد که مشتریان کسب و کار ها در یک واحد می بینند.

3. ابتکاری که سرمایه داران و یا صاحبان پول آن را یک بودجه واحد می بینند.

از لحاظ توسعه نرم افزار سه دسته اصلی اپلیکیشن وجود دارد.

**Native Apps**

این دسته از اپلیکیشن ها مختص یک سیستم عامل مثلا اندروید یا IOS نوشته می شوند.بدیهی است که نمیتوان از این دست اپلیکیشن ها در هر سیستم عاملی استفاده کرد به عنوان مثال نمیتوان از اپلیکیشن اندروید در IOS استفاده نمود با این حال، توسعه دهنده میتواند به طور وسیع از لیست زبان های برنامه نویسی، زبان مورد نظر خود را گزینش و شروع به توسعه نرم افزار خود کند. از فواید این گونه اپلیکیشن ها میتوان به Performance بالای آن ها اشاره کرد. علاوه بر آن، UI و UX درخور و حرفه ای، دسترسی به امکانات مختلف Gadget مورد نظر مانند Bluetooth و GPS را در بر می گیرد.

اما Native App ها دارای بعد منفی نیز هستند که مهمترین آن میتوان به علاقه افراد صاحب کسب و کار به نرم افزار هایی در سیستم عامل های مختلف جوابگوی نیاز های آن ها باشد، ولی این دسته از اپلیکیشن ها این کار را انجام نمی دهند. از طرفی این گونه اپلیکیشن ها نیازمند فضای ذخیره سازی بیشتری هستندبه خصوص برای دریافت نسخه های جدید تر بروزرسانی ها.

**Hybrid Apps**

اینستاگرام و Gmail از محبوب ترین اپلیکیشن های این دسته به شمار می روند. این دسته از اپلیکیشن ها یک Web App هستند که نقش یک Natice App را ایفا می کنند و در تمامی سیستم عامل ها قابل اجرا هستند. این گونه اپلیکیشن ها یک کد مرکزی دارند که میتوان از آن خروجی Hybrid App نیز گرفت. در نتیجه سرعت توسعه در این نرم افزار ها بالا است. همچنین Hybrid App ها میتوانند یک نسخه MVP از نرم افزار به عنوان یک نسخه تستی برای نسخه Native آماده کند.

از ویژگی های منفی Hybrid App ها میتوان به Performance ضعیف نسبت به همتایان Native اشاره کرد.

**Web Apps**

این گونه اپلیکیشن ها در مرورگر ها اجرا می شوند و در همه دستگاه های دارای سیستم عامل و مرورگر بدون دانلود هیچ گونه فایلی قابل اجرا هستند. این گونه اپلیکیشن ها از دوبخش تشکیل می شوند: 1. FrontEnd که از زبان های به خصوصی مانند Html,css,js,..... ساخته می شوند 2. BackEnd که از زبان های به خصوصی مانندjava,c#,js,.... ساخته می شوند.به هیچ گونه فضای ذخیره سازی از سمت کاربر نیازی نداردوطراحی کاربری آن ها با هر دستگاه و گجتی همخوانی دارد.

در بعد منفی این گونه نرم افزار ها میتوان به همگام نبودن یک سری تکنولوژی های توسعه با مرورگر های مختلف،ناتوانی در اجرای اپلیکیشن در صورت آفلاین بودن،وابسته بودن به اپلیکیشن جانبی مانند مرورگر

**معماری نرم افزار چیست؟**

بحث های زیادی بابت تعریف پایه ای از اینکه معماری نرم افزار چیست شده اما در این تعریف به نظرات آقای Martin Fowler و Ralph Johnsonدو معمار مطرح در زمینه معماری نرم افزار، می پردازیم.

همانطور که پیشتر به آن اشاره کردیم تعریف واحدی از معماری نرم افزار وجود ندارد، اما عموم مردم گمان می کنند که معماری نرم افزار چیزی شبیه به سازماندهی اساسی یک سیستم است یا طرز اتصال ماژول های سطح بالا برای ارتباط به هم. اما طبق طرز تفکر Martin Fowler که بر اساس رایزنی هایی که با Ralph Johnson داشته شکل گرفته و این عبارات زیر سوال برده شده،هیچ تعریفی عینی برای ماژول های سطح بالا و یا اساس یک سیستم نرم افزاری وجود ندارد.

حالت دوم تعریف که عموم مردم با آن سروکار دارند اینست که"تصمیماتی برای طراحی که در ابتدای پروژه گرفته می شود." با این تعریف دوباره آقای Ralph Johnson مخالفت خودش را با این تعریف نیز مطرح نمود با این عنوان که این جمله بیشتر شبیه به این است که تصمیماتی که آرزو دارید در ابتدای پروژه بگیرید.

درنتیجه گیری که Ralph Johnson در جستجوی تعریفی برای معماری نرم افزار کرد این بود که معماری در رابطه با چیز های مهم است، هر چیزی که باشد. Martin Fowler راجع به این حرف Ralph Johnson می گوید که در نگاه اول، به نظر عادی می رسد، اما این عبارت بار معنایی فراوانی را به دوش می کشد.این به ین معنی است که در قلب تفکر معماری وار(معماری نرم افزار) اینست که تصمیم بگیریم چه چیزی در نرم افزار مورد نظر برای ما مهم است که سپس انرژی خود را برای نگه داشتن آن بخش مورد نظر صرف کنیم.برای توسعه دهنده ای که قصد دارد به یک معمار نرم افزار تبدیل گردد، به این موضوع نیاز دارد که به توانایی برسد که بتواند عناصر مهم و کلیدی نرم افزار را تشخیص دهد و تشخیص دهد که چه بخش هایی در صورت نداشتن کنترل روی آن، سیستم را دچار مشکل خواهد کرد.

**چرا معماری اهمیت دارد؟**

معماری نرم افزار یک موضوع گمراه کننده برای مشتریان نرم افزار و کاربران آن می باشد،چرا که معماری، آن چیزی نیست که عموم مردم گمان می کنند.معماری ضعیف یک نرم افزار شریک جرم افزایش بی نظمی در نرم افزار است و این بی نظمی، مانع توانایی توسعه دهندگان در درک پروژه می باشد.

بی نظمی زیاد در پروژه موجب سختی زیاد در تغییر و کندی در توسعه ویژگی های جدید در پروژه می باشد.

عموما عادت کرده ایم که در پروژه های خود کیفت بالا را فدای هزینه های زیاد پروژه می کنیم. این رویکرد در مواردی مانند طراحی رابط های کاربری جوابگو است، اما وقتی وارد معماری نرم افزار و دیگر جنبه های داخلی نرم افزار می شویم، این رابطه عکس می شود. **کیفیت بالای بخش داخلی نرم افزار اضافه شدن ویژگی ها و امکانات جدید به نرم افزار سرعت می بخشد،** که دلیل آن بی نظمی حداقلی در نرم افزار است و این موضوع چیزی نیست که با ابزاری یا به طور عینی اندازه گیری شود یا به عبارت دیگر مقیاس پذیر نیست بلکه توسعه دهنده با تجربه در هفته یا در ماه ها به این نتیجه خواهد رسید.

**طبقه بندی سبک های معماری**

**1. Data Centered Archtectures**

منبع داده ها در مرکز این معماری قرار دارد و مکررا برای ایجاد تغییراتی مانند عملیات Add، Update، Delete، modifyتوسط بخش های دیگر در دسترس خواهد بود.این سبک یکپارچگی را در نرم افزار گسترش می دهد، به طوری که بخش های موجود قابل تغییر و یا بخش های جدیدی قابل اضافه شدن بدون دسترسی خاصی از سمت کاربر یا بدون کوچکترین ارتباطی با کاربر، امکان پذیر خواهد بود.داده میتوانند توسط مکانیزم Blackboard در اختیار کاربر قرار گیرد.

**مزایا:**

* مخزن اطلاعات به طور مستقل در اختیار کاربر قرار می گیرد.
* کاربران به طور مستقل از هم کار می کنند.
* ایجاد تغییر به آسانی صورت می گیرد.

**2. Data Flow Architectures**

در این سبک داده ورودی و طی فرایندی به داده خروجی تبدیل می شود که طی این فرایند، دستکاری های محاسبتی و یک سری اعمال و کارها روی داده ها صورت می گیرد. در صورتی که جریان داده ها به یک خط تبدیل، تبدیل گردند، دنباله ای تشکیل می شود که به عنوان دنباله ترتیبی شناخته می شود. این ساختار دسته ای از داده ها را می پذیرد و سپس یک سری اجزای متوالی را برای تبدیل آن اعمال می کند.