عنوان مقاله: معماری مونولیتیک

نویسنده مقاله: تیم فنی نیک آموز

تاریخ انتشار: ۱۳ مهر ۱۴۰۲

منبع: https://nikamooz.com/monolithic-architecture

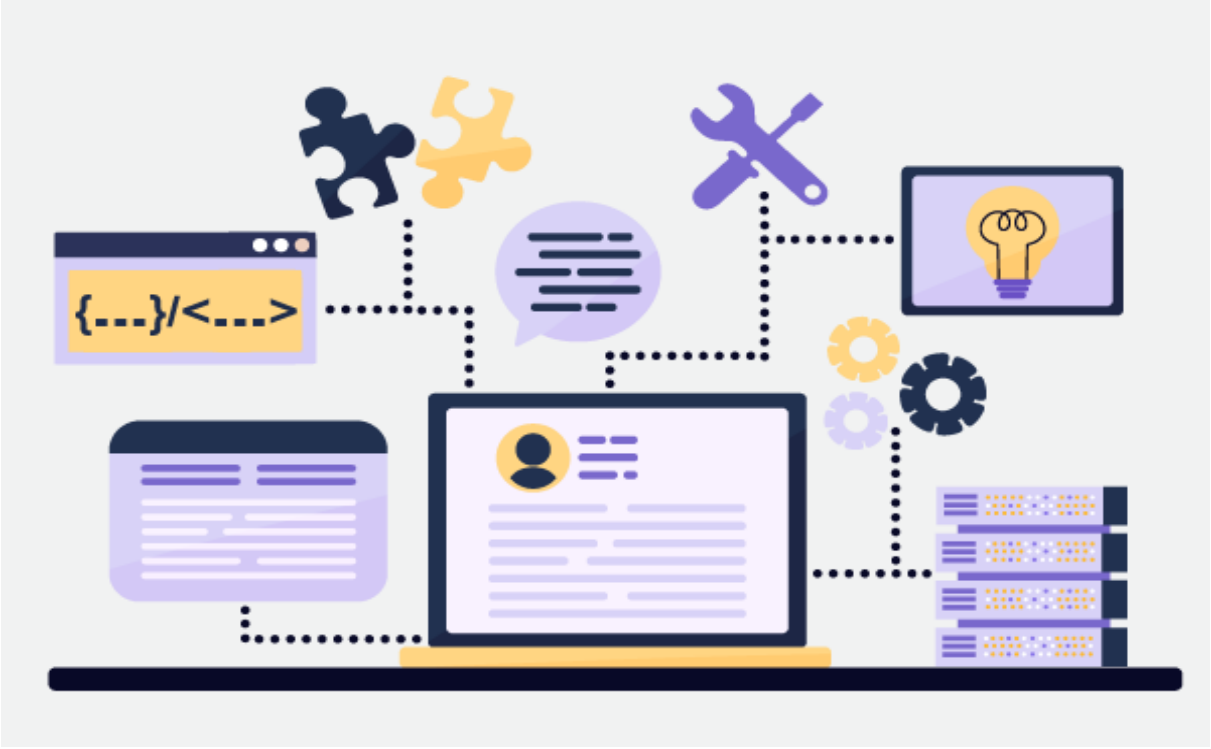
معماری مونولیتیک یک الگوی طراحی است که در آن، برنامه به‌صورت یک واحد کامل پیاده‌سازی می‌شود. معماری یکی از مفاهیم بنیادی در حوزه توسعه نرم‌افزار (Software Development) است که تمام اپلیکیشن‌ها براساس آن ساخته می‌شوند. یکی از رویکردهای سنتی آن، معماری مونولیتیک (Monolithic Architecture) است که قصد داریم در این مقاله، آن را مورد بررسی قرار دهیم. در ادامه نیز مزیت‌ها و معایب معماری Monolithic را همراه با مثال‌های مختلف شرح می‌دهیم تا مشخص شود آیا این معماری، برای پروژه نرم‌افزار شما مناسب است یا خیر. پیش از پرداختن به این مفهوم، لازم است به تعریف معماری نرم افزار بپردازیم.

معماری نرم افزار چیست؟

معماری نرم‌افزار (Software architecture) یک ساختار سطح بالا یا اصطلاحاً «نقشه ساخت» (Blueprint) است که بااستفاده از آن، نحوه طراحی و سازماندهی سیستم نرم‌افزار تعریف می‌شود. معماری نرم افزار به‌عنوان پایه اصلی برای مواردی همچون توسعه، نگهداری (Maintenance) و مقیاس‌گذاری (Scaling) اپلیکیشن‌های نرم‌افزاری محسوب می‌شود. در ادامه، برخی از جنبه‌های اصلی معماری نرم افزار را شرح می‌دهیم تا آن را بهتر درک کنید:

- **اجزا و ماژول‌ها:** معماری نرم‌افزار، اجزا (Components) و ماژول‌های (Modules) اصلی یک سیستم و نحوه تعامل آن‌ها با یکدیگر را تعریف می‌کند. باید توجه داشت که این اجزا می‌توانند کلاس‌ها (Classes)، توابع و حتی خدمات مستقل از هم باشند.
- **الگوی طراحی (Design Pattern):** معمولاً معماری نرم‌افزار با دیزاین پترن به‌صورت ترکیبی کاربرد دارند. دیزاین پترن راه حل‌هایی هستند که به مشکلات رایج طراحی نرم‌افزار پاسخ می‌دهند. این الگوها کمک می‌کنند تا ساختار نرم‌افزار به‌گونه‌ای باشد که قابلیت استفاده مجدد (Reusability)، امکان نگهداری (Maintainability) و انعطاف‌پذیری آن افزایش پیدا کند.
- **صفت‌های کیفی (Quality Attributes):** معماری نرم‌افزار، صفت‌های کیفی مختلفی مانند کارایی (Performance)، امنیت (Security)، قابل اطمینان بودن (Reliability)، مقیاس‌پذیری (Scalability) و قابلیت نگهداری را در بر می‌گیرد. از این رو، معماری نرم‌افزار باید به گونه‌ای طراحی شود که این نیازمندی‌ها را پوشش دهد.
- **تصمیم‌گیری و Trade-off:** به‌واسطه معماری، تصمیمات حیاتی متعددی درخصوص انتخاب‌های مربوط به تکنولوژی، ذخیره‌سازی داده‌ها (Data Storing)، پروتکل‌های ارتباطی و سایر موارد اتخاذ می‌شود. در اغلب موارد، این تصمیم‌ها یک Trade-off یا اصطلاحاً «انتخاب‌های چند معیاره» بین اهداف هستند.

- **معماری لایه‌ای (Layered) یا پلکانی (Tiered):** سیستم‌های نرم‌افزاری مختلفی وجود دارند که معماری لایه‌لایه یا پلکانی دارند و هر یک از لایه‌هایشان، کارایی مخصوص به خود را دارند. به‌عنوان مثال، ممکن است یک وب‌اپلیکیشن، لایه‌های مختلفی مانند لایه ارائه (Presentation)، لایه منطق کسب و کار (Business Logic) و لایه دسترسی به داده‌ها (Data Access) را داشته باشد.
- **مدل‌های Client-Server:** معمولاً معماری‌ها شامل مدل‌های سرور – کلاینت هستند که در آن‌ها، کلاینت‌ها با سرورها در ارتباط‌اند و این موضوع می‌تواند به‌صورت متمرکز (Centralized) یا توزیع‌شده (distributed) روی چند سرور باشد.
- **میکروسرویس و مونولیت:** معماری نرم‌افزار تعیین‌کننده این است که سیستم معماری مونولیت (اپلیکیشن به‌صورت واحد و کاملاً یکپارچه) یا مبتنی‌بر میکروسرویس (ترکیبی از سرویس‌های کوچک و مستقل) ساخته شود. باید توجه کرد که این تصمیم حیاتی، برروی مقیاس‌پذیری و قابلیت نگهداری مؤثر خواهد بود.
- **مستندات (Documentation):** مستندات جامع و مفصل، یکی از موارد ضروری در معماری نرم‌افزار است؛ زیرا با کمک آن، توسعه‌دهندگان، ساختار سیستم و قوانین طراحی را بهتر درک می‌کنند و به‌دنبال آن، کار کردن برروی Codebase آسان‌تر می‌شود.
- **نگهداری و سیر تکاملی تدریجی:** لازم است معماری نرم‌افزار بتواند با تغییرات و به‌روزرسانی‌های گوناگون، خود را تطبیق دهد. در حقیقت، یک معماری مطلوب به ما اجازه می‌دهد بدون اینکه از پایه آن را مجدداً طراحی کنیم، تنها به‌صورت افزایشی بهبودش دهیم.
- **ابزارها و فریم ورک‌ها:** در اغلب موارد، معماری از انواع چارچوب (Framework)، ابزارها و روش‌های توسعه مختلف برای پیاده‌سازی مؤثر طراحی معماری استفاده می‌کند.



انواع معماری نرم افزار

در این بخش، برخی از انواع معماری نرم افزار رایج را معرفی می‌کنیم:

- معماری لایه‌ای (Layered)
- کلاینت – سرور (Client – Server)
- ارباب – برده (Master – Slave)
- معماری Pipe – Filter
- معماری کارگزار (Broker)
- الگوی معماری نظیربه‌نظیر (Peer To Peer)
- الگوی Event – Bus
- معماری Model – View – Controller
- الگوی تخته‌سیاه (BlackBoard)
- الگوی interpreter

تا این بخش از مقاله، معماری نرم افزار، انواع و جوانب مختلف آن بررسی شده‌اند. حال با این مقدمه می‌خواهیم به این سؤال پاسخ دهیم که معماری مونولیتیک (Monolithic) چیست؟

معماری مونولیتیک (Monolithic) چیست؟

همان‌طور که از نام معماری مونولیتیک مشخص است، به ساختن یک نرم‌افزار یکه و یکپارچه گفته می‌شود که در آن، تمام اجزا با یکدیگر ارتباط تنگاتنگی دارند. به بیان ساده، ایجاد معماری Monolithic مانند ساختن یک ساختمان بزرگ و واحد است که هر یک از اتاق‌های آن، دارای هدف مشخصی هستند و منابع مشترکی را مورد استفاده قرار می‌دهند.

مثال برای معماری مونولیتیک

- **پلتفرم‌های تجارت الکترونیکی:** وب سایت‌های تجارت الکترونیک (E-Commerce) متعددی از معماری مونولیتیک استفاده می‌کنند. در چنین پلتفرم‌هایی، کارت خرید، کاتالوگ محصول و فرآیند پردازش، همگی درون یک کد بیس (Code Base) واحد، ارتباط تنگاتنگی با یکدیگر دارند.
- **سیستم‌های مدیریت محتوا (CMS):** در نسخه‌های نخستین پلتفرم‌هایی مانند وردپرس (WordPress)، از معماری مونولیتیک استفاده شده است.
- **سیستم‌های برنامه‌ریزی منابع سازمان (ERP):** معمولاً سازمان‌های بزرگ از سیستم‌های ERP مونولیتیک استفاده می‌کنند که در آن‌ها، ماژول‌های مدیریت امور مالی، منابع انسانی و موجودی، همگی باهم به‌صورت یکپارچه به یک اپلیکیشن تبدیل شده‌اند.
-

مزایای معماری مونولیتیک چیست؟

معماری مونولیتیک دارای مزیت‌هایی است که در این بخش به آن می‌پردازیم:

- **سادگی:** به‌طور کلی، توسعه و نگهداری سیستم‌های مونولیتیک آسان‌تر است؛ چراکه ساختار آن قابل درک و سرراست است.
- **استقرار به‌صورت یکه و واحد:** تمام اپلیکیشن به‌صورت «یک واحد» استقرار یا اصطلاحاً «دیپلوی» (Deploy) می‌شود. از این رو، راه‌اندازی آن نرم‌افزار ساده‌سازی می‌شود و مشکلات مربوط به استقرار کاهش می‌یابد.
- **خروجی سریع:** استفاده از معماری مونولیتیک شما را سریع‌تر به خروجی می‌رساند؛ اما یکی از چالش های معماری میکروسرویس این است که نیازمند زمان بیشتری است.
- **افزایش میزان کار مفید انجام شده توسط برنامه‌نویس:** معماری Monolithic به‌گونه‌ای است که بازدهی توسعه‌دهندگان بیشتر می‌شود؛ زیرا آن‌ها می‌توانند بدون نگرانی درمورد ارتباط میان میکروسرویس‌ها، روی قسمت‌های مختلف اپلیکیشن کار کنند.

معایب معماری مونولیتیک چیست؟

برخی از معایب معماری Monolithic عبارتند از:

- **مقیاس‌پذیری (Scalability):** مقیاس‌گذاری سیستمی که با معماری مونولیتیک ایجاد شده است، می‌تواند با چالش همراه باشد؛ چراکه برای مقیاس‌گذاری آن، باید این عمل روی تمام اپلیکیشن انجام شود. حتی در شرایطی که تنها یک جز از سیستم به منابع بیشتر نیاز داشته باشد، این شرایط صادق است.
- **اندازه کد بیس:** هرچه اپلیکیشن گسترده‌تر می‌شود، Codebase آن پیچیده‌تر خواهد شد و به‌دنبال آن، درک چنین سیستمی با چالش همراه است.
- **محدودیت در انتخاب‌های تکنولوژی در دسترس:** ممکن است به‌دلیل انتخاب یک تکنولوژی مشخص در شروع پروژه، مجبور به ادامه دادن آن با تکنولوژی‌ها و فریم ورک‌های قدیمی شویم. طبیعتاً این موضوع می‌تواند در برخی موارد بازدارنده باشد.
- **ریسک Downtime:** باتوجه به ساختار یکپارچه و واحد در سیستم مونولیتیک، درصورت بروز خطا یا قطعی در یک بخش از سیستم، ممکن است روی تمام سیستم اثر بگذارد و به‌دنبال آن، قطعی یا Downtime ایجاد شود.
- **نگهداری و توسعه دشوار:** باتوجه به اینکه سیستم به‌صورت یکپارچه است و در طول زمان گسترش داده می‌شود، نگهداری و توسعه آن پیچیده خواهد بود.

چه زمانی باید سراغ معماری مونولیتیک برویم؟

درنهایت، می‌توان نتیجه گرفت که معماری مونولیتیک برای توسعه نرم‌افزارهای کوچک مناسب است و در این شرایط، مشکلی ایجاد نمی‌کند؛ بنابراین، در صورتی که نیازمندی‌ها و همچنین اندازه نرم‌افزار شما گسترده نیست، این معماری گزینه مطلوبی برای شما است و لازم نیست به‌سراغ میکروسرویس بروید.

ممکن است با خواندن چالش‌های ذکر شده در این بخش از مقاله معماری مونولیتیک چیست، ترجیح دهید راه دیگری را در پیش بگیرید. در چنین شرایطی، معماری میکروسرویس می‌تواند گزینه جایگزین باشد. برخلاف معماری Monolithic، معماری میکروسرویس هر یک از بخش‌های کاربردی اپلیکیشن را به‌عنوان یک سرویس مستقل درنظر می‌گیرد و به همین دلیل، می‌توان هر یک از این بخش‌های مستقل را تغییر داد، به‌روزرسانی کرد و حتی کامل از سیستم حذف کرد.

جمع بندی

به‌طور کلی، معماری مونولیتیک مزیت‌هایی همچون سادگی و افزایش کارایی را دارد؛ با این حال، این معماری با چالش‌های مربوط به مقیاس‌پذیری و افزایش اندازه کدبیس همراه است. زمانی که می‌خواهید برای معماری نرم‌افزار تصمیم‌گیری کنید، لازم است پتانسیل گسترش و نیازمندی‌های خاص پروژه خود را به‌طور دقیق مورد تجزیه و تحلیل قرار دهید. هرچند ممکن است معماری مونولیتیک برای برخی اپلیکیشن‌ها مفید واقع شود، اما استفاده از رویکردهای مدرن مانند میکروسرویس‌ها، گزینه مطلوب محسوب می‌شود. بنابراین، با در نظر داشتن تمام جوانب مسئله، معماری نرم‌افزار خود را به‌صورت آگاهانه انتخاب کنید.