هفت خان توسعه بک‌اند

خان صفرم و خان اول: سفری به قلمرو بک‌اند

رضا قمی‌الاصل
دانشجو مهندسی کامپیوتر
دانشکدگان فارابی دانشگاه تهران
rezaqomyalasl@gmail.com

۱۲ دقیقه

سلام و درود بی‌کران بر شما مسافران جاده دانش!

توی این مقاله و سری‌های بعدی، می‌خوایم ببینیم راه و رسم «بک‌اندی شدن» چیه و چطوریه. این سفر، مسیر کساییه که می‌خوان مغز متفکر و موتور محرک یه نرم‌افزار باشن.

برای اینکه توی این مسیر طولانی و پرماجرا گم نشیم، یک نقشه‌راه جامع و مرحله‌به‌مرحله با عنوان «هفت خان» آماده کردیم. قراره قدم به قدم از صفر شروع کنیم، بریم جلو و از مفاهیم اولیه گرفته تا مقیاس‌پذیری و بازار کار رو فتح کنیم. این مسیر توی چهار قسمت مجزا (چهار مقاله) با شما همراه خواهد بود:

نقشه‌راه هفت‌خانِ توسعه‌دهنده بک‌اند

قسمت اول (همین مقاله): دروازه‌ی ورود

خان صفرم: ورود به قلمرو پنهان (مفاهیم پایه‌ای که برای شروع این سفر بهشون نیاز داریم)

خان اول: آماده‌سازی برای نبرد (فریم‌ورک‌ها، دیتابیس رابطه‌ای و ابزارهای ضروری)

قسمت دوم: عمق و استحکام

خان دوم: غواصی در اقیانوس داده‌ها (تسلط عمیق بر پایگاه‌های داده، NoSQL و Caching)

خان سوم: نگهبانان و قفل‌های آهنین (اصول امنیت، احراز هویت و طراحی API های استاندارد)

قسمت سوم: مهندسی یک سیستم بزرگ

خان چهارم: معماران بناهای ماندگار(معماری نرم‌افزار، اصول SOLID و دیزاین پترن‌ها)

خان پنجم: رام کردن هیولای مقیاس (میکروسرویس‌ها، صف‌ها و مدیریت ترافیک‌های بالا)

قسمت چهارم: رسیدن به مقصد

خان ششم: فرماندهی لشکر سرورها (ابزارهای DevOps، استقرار کد و مانیتورینگ سیستم)

خان هفتم: بازار کار و گنجینه مهارت‌ها (رزومه، مصاحبه و مهارت‌های نرم لازم برای موفقیت)

اما خب، قبل از اینکه بزنیم به دل جاده و وارد خان صفرم بشیم، میخوایم بگیم که بک‌اند اصلاً چیه و چرا تا این حد برای هسته‌ی هر نرم‌افزاری اهمیت داره.

خان صفرم: ورود به قلمرو پنهان

(بک‌اند چیست و چرا مهم است؟)

این خان، دروازه ورود به سفر است؛ جایی که با مفاهیم بنیادی دنیای سرور آشنا می‌شویم و ابزارهای اولیه خود را برمی‌گزینیم.

پرده‌برداری از قلمرو نهان (بک‌اند چیه؟)

(ماجرای پشت پرده نرم‌افزارها)

فرض کنید که می‌خواید از نرم‌افزاری مثل آمازون، دیجی‌کالا یا هر نرم‌افزار دیگه‌ای که توی وب هست، استفاده کنید. وقتی وارد سایت می‌شید، یه صفحه می‌بینید که داخلش یک‌سری عکس، منو، دکمه و این‌ها وجود داره که شما می‌تونید توی سایت باهاشون بگردید و سایت رو بالا و پایین کنید. این بخشیه که شما به عنوان یه کاربر مستقیماً باهاش سروکله می‌زنید. ما به اون بخش می‌گیم Client-Side یا همون فرانت‌اند؛ یعنی جایی که کاربر باهاش کار می‌کنه و تمام اتفاقات بصری اونجا رخ می‌ده. کدهای فرانت‌اند توی مرورگر شما اجرا می‌شن و کارشون اینه که داده‌ها رو خوشگل کنن و نشون بدن. البته حواستون باشه فرانت فقط مرورگر نیست؛ می‌تونه نرم‌افزاری باشه که مستقیم روی گوشیتون نصب می‌کنید یا حتی دستگاه عابر بانکی باشه که ازش پول می‌گیرید.

خب، جدای از این بخش (فرانت‌اند)، یه قسمت دیگه هم داریم که تمام کارِ فکری و محاسباتی رو انجام می‌ده. مثلاً فرانت‌اند لیست محصولات سایت، اینکه امتیازش چقدره، چقدر موجودی داره، قیمتش چنده، تخفیف داره یا نداره رو از یه جایی می‌گیره. این اطلاعات نه تنها باید یه جا ذخیره بشن، بلکه باید قبل از نمایش، منطق‌های پیچیده‌ای هم روشون اعمال بشه؛ مثلاً بررسی بشه که آیا کاربر اصلاً اجازه دیدن این قیمت رو داره یا نه، یا اگه یه تخفیف خاص خورده، محاسبه قیمتش چطوریه. تمام این کارها باید سمت یه سرور هندل بشن که به این بخش ما می‌گیم Server-Side یا همون بک‌اند خودمون.

بک‌اند در واقع سه وظیفه خیلی مهم داره:

مدیریت داده: ذخیره، سازماندهی و بازیابی تمام اطلاعات حیاتی مثل رمز عبور کاربران (البته به صورت هش شده)، لیست سفارشات، مشخصات محصولات و...

اجرای منطق کسب‌وکار(Business Logic): هسته اصلی نرم‌افزار، یعنی قوانین و فرآیندهای کسب‌وکار (مثلاً اینکه چطوری پول از حساب کاربر کم بشه، یا چطور یک سفارش تأیید بشه).

تأمین امنیت: مطمئن شدن از اینکه هر کسی فقط به اطلاعاتی دسترسی پیدا کنه که حقشه و جلوگیری از حمله‌های مختلف سایبری.

حالا که فرق این دو دنیا رو حسابی فهمیدیم، بریم و سیر و سلوک بک‌اندی شدن رو با هم شروع کنیم.

آموختنِ زبانِ ارتباط (آشنایی با شبکه)

(آشنایی با زبان مشترک اینترنت)

شبکه، اولین چیزیه که باید در موردش بدونیم. دلیلش هم واضحه: بک‌اند کلاً روی شبکه‌ست و باید زبان ارتباطات رو بدونه.

البته قرار نیست توی بک‌اند یه شبکه کار حرفه‌ای بشید (مثل متخصصانی که کانفیگ روتر و سوئیچ انجام می‌دن). فقط لازمه که اولش یه دانش پایه محکم از شبکه داشته باشید که اگه بعداً به مشکلی چیزی برخوردید، مثلاً یه درخواست تایم‌آوت شد، بتونید گلیم خودتون رو از آب بکشید بیرون. این دانش به شما این دید رو می‌ده که بفهمید داده‌ها چطور توی اینترنت جابه‌جا می‌شن.

برای شبکه، اگه در این حد بدونید که IP چیه (که مثل آدرس منحصربه‌فرد دستگاه شما توی شبکه‌ست)، Port چیه (که مثل شماره ورودی یک سرویس خاص روی اون آدرسه)، DNS چطوری اسم سایت‌ها رو به همون آدرس‌های IP ترجمه می‌کنه، فرق TCP با UDP چیه و مفاهیم پایه این شکلی، کافیه.

انتخاب قلب تپنده (زبان برنامه‌نویسی)

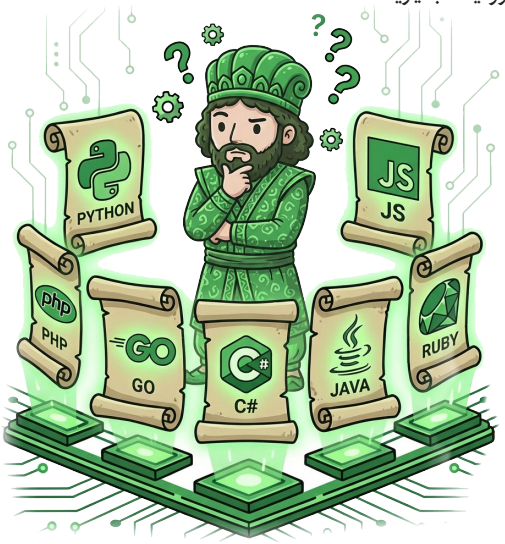
(یادگیری فن کدنویسی و اصول مهندسی)

بعد از اینکه یکم دستتون تو شبکه گرم شد و اصولش رو فهمیدید، اینجا نوبت انتخاب یه زبان برنامه‌نویسیه که باهاش کدتون رو پیاده‌سازی کنید.

زبان‌های برنامه‌نویسی زیادی سمت بک‌اند وجود دارن و استفاده می‌شن. هرکدوم یه عالمه مزایا و معایب دارن که بسته به نوع پروژه باید یکی رو انتخاب کرد. مثلاً Ruby، Java، C#، PHP، Node.js، Go، Python و تعداد فراوانی از زبان‌های برنامه‌نویسی دیگه، در این میدان حضور دارن.

نکته مهم اینه که: شما باید یکیش رو انتخاب کنید.

اگه از قبل هرکدوم از این زبان‌ها رو بلدید یا باهاش کار کردید، می‌تونید همون رو انتخاب کنید. این خیلی خوبه که از نقطه‌ای شروع کنید که توش قوی هستید. اما اگه تا الان با هیچ‌کدوم از زبان‌های برنامه‌نویسی کار نکردید، پیشنهاد من اینه که با Python شروع کنید و مفاهیمش رو یاد بگیرید. پایتون به خاطر خوانایی بالا و سادگی، شما رو درگیر پیچیدگی‌های زبانی نمی‌کنه و می‌تونید سریع‌تر مفاهیم اصلی بک‌اند رو یاد بگیرید.



فارغ از اینکه چه زبانی رو انتخاب می‌کنید، باید روی مفاهیم اصلی اون زبان مسلط بشید:

ساختارهای داده(Data Structures): اینکه لیست، دیکشنری، آرایه، درخت یا گراف چیه و هرکدوم کجا به کار میان.

الگوریتم‌ها(Algorithms): چطوری کدی بنویسیم که کارآمد باشه و سرعتش بالا باشه (مثلاً چطوری یه لیست رو جستجو یا مرتب کنیم).

مفاهیم شیءگرایی(OOP): اینکه کلاس و شیء و وراثت چیه و چطوری کدمون رو تمیز و ماژولار بنویسیم.

همزمانی(Concurrency & Multi-threading): این مفهوم شاید کمی پیچیده باشه اما حیاتیه. بک‌اند برخلاف کدهای معمولی که خط به خط اجرا می‌شن، باید بتونه همزمان به صدها یا هزاران کاربر جواب بده. باید یاد بگیرید چطور برنامه‌ای بنویسید که وقتی یک کاربر منتظر جواب دیتابیسه، برنامه متوقف نشه و بتونه کار بقیه کاربرا رو راه بندازه (مفهوم Blocking vs Non-Blocking).

این مفاهیم، هسته کار هر مهندس نرم‌افزار رو تشکیل می‌دن، نه فقط بک‌اند دولوپر.

زبان مشترک بک‌اند و فرانت‌اند (پروتکل HTTP)

(زبان ارتباطی بک‌اند و فرانت‌اند)

بعد از انتخاب زبان برنامه‌نویسی، باید برید و یکم در مورد HTTP (Hypertext Transfer Protocol) بدونید. گفتیم که این پروتکل، زبان گفتگوی وب و اساس کار بک‌اند ماست. شما باید بفهمید که سرور شما چطوری درخواست‌ها رو دریافت می‌کنه و بهشون پاسخ می‌ده.

این درک شامل:

متدها(Methods): چه متدهایی داره؟ مثلاً GET (برای دریافت داده)، POST (برای ارسال و ساختن داده)، PUT و PATCH (برای ویرایش داده) و DELETE (برای پاک کردن داده). هرکدوم از این‌ها باید دقیقاً برای کاری که تعریف شدن، استفاده بشن.

هدر(Header): هدر در واقع اطلاعات اضافی و متاداده‌ای درخواست رو حمل می‌کنه؛ مثل نوع محتوایی که فرستاده می‌شه (مثلاً JSON، توکن‌های امنیتی یا کوکی‌ها).

بادی(Body): همون محتوای اصلی داده‌ای که فرستاده یا دریافت می‌شه. مثلاً وقتی کاربر توی یه فرم ثبت نام می‌کنه، اطلاعاتش توی بادی درخواست POST قرار می‌گیره.

کدهای وضعیت (Status Codes): باید بدونید که کدهای وضعیت مثل ۲۰۰ (موفقیت‌آمیز)، ۴۰۴ (پیدا نشد)، ۴۰۱ (مجوز نداری) و ۵۰۰ (خطای سرور) چیا هستن و کی باید از کدوم استفاده کرد. استفاده درست از این کدها، به فرانت‌اند و کلاینت کمک می‌کنه که بفهمه عملیات شما موفق بوده یا شکست خورده و دلیل شکست چی بوده.

کوکی(Cookie): کوکی‌ها تکه‌های کوچکی از داده هستن که سرور برای ذخیره وضعیت کاربر (مثلاً اینکه کی وارد شده) در مرورگر کلاینت نگه می‌داره.

درک این ساختار پروتکل، به شما کمک می‌کنه که بتونید باگ‌ها رو بهتر پیدا کنید و APIهای (رابط‌های برنامه‌نویسی) تمیزتری طراحی کنید.

البته این رو هم در نظر داشته باشید که HTTP تنها پروتکلی نیست که نیازه؛ پروتکل‌های دیگه‌ای هم هستن که هر کدوم در جای خودشون کاربرد دارن و در قسمت‌های بعدی معرفیشون می‌کنیم.

خان اول: آماده‌سازی برای نبرد

در این خان، با ابزارهای قدرتمندی که برای ساخت، سازماندهی و نگهداری نرم‌افزار لازم است، آشنا می‌شویم.

زره و سلاحِ نبرد (فریم‌ورک‌ها و لایبری‌ها)

(استفاده از فریم‌ورک‌های توسعه سریع)

بعد از اون، باید برید سراغ یک فریم‌ورک یا لایبرری داخل همون زبان برنامه‌نویسی که انتخاب کردید.

در واقع، اون فریم‌ورک یا لایبرری یه لایه میانجی هست بین اون زبان برنامه‌نویسی که انتخاب کردید و پروتکل HTTP. این ابزارها، کارهای تکراری و پیچیده رو قبلاً برای شما نوشتن؛ مثلاً سیستم مسیریابی درخواست‌ها، مدیریت خطاها یا امنیت اولیه. شما با استفاده از این ابزارها می‌تونید سریع‌تر بک‌اند خودتون رو توسعه بدید و تمرکزتون رو بذارید روی منطق اصلی کسب‌وکار.

زبان‌های برنامه‌نویسی مختلف، فریم‌ورک‌ها و لایبرری‌های متفاوتی دارن:

پایتون: FastAPI (خیلی سریع و مدرن)، Flask (ساده و مینیمال)، Django (کامل و جامع)

Go: Gin، Echo، Fiber

Node.js: Express، NestJS

نکته کلیدی: نکته مهمی که وجود داره و باید حسابی تو ذهنتون باشه، اینه که: شما نباید سعی بکنید یک فریم‌ورکر بشوید. یعنی برای مثال جنگو رو یاد بگیرید و اسم خودتون رو بذارید "جنگو دولوپر" و فقط کد جنگو بزنید. شما باید به یه مهندس نرم‌افزار (Software Engineer) تبدیل بشید؛ یعنی کسی که به مفاهیم تسلط داره و می‌تونه با ابزارهای موجود، یک نرم‌افزار رو توسعه بده، نه کسی که فقط یک ابزار رو بلده. اگر فردا روزی پروژه شما نیاز به زبان یا فریم‌ورک دیگه‌ای داشت، باید توانایی این رو داشته باشید که اون ابزار رو یاد بگیرید و باهاش کار کنید. پس روی ابزارتون تعصب نداشته باشید و همیشه ذهنتون رو برای یادگیری ابزارهای جدید باز نگه دارید.

خزانه‌داریِ اسناد (دیتابیس‌ها و SQL)

(مدیریت داده‌های نرم‌افزار)

هر نرم‌افزار بک‌اندی نیاز به یک‌سری اطلاعات داره که باید یه جا ذخیره بشن؛ مثلاً لیست اسامی یوزرها با هش پسوردشون، تاریخی که عضو سایت شدن، لیست محصولات یا موجودی کیف پول کاربران. این اطلاعات قراره توسط یه دیتابیس(Database) ذخیره و مدیریت بشن.

ما انواع مختلفی از دیتابیس‌ها رو داریم، ولی پرکاربردترین اون‌ها، دیتابیس‌های رابطه‌ای(Relational Databases) هستن. در این مدل، شما اطلاعات رو توی یه جدول ذخیره می‌کنید. مثلاً یه جدول داریم به اسم users که یک‌سری ستون داره (مثلاً created_at، email، name و...) و هر یوزر یه ردیف جدید داخل این ستون داره. این جداول با منطق مشخصی به هم ارتباط دارن. این نوع دیتابیس برای داده‌هایی که ساختار مشخص و روابط ثابتی دارن (مثل تراکنش‌های مالی یا مشخصات کاربر)، فوق‌العاده مطمئن و مناسبه.

یه زبان برنامه‌نویسی وجود داره به اسم SQL (Structured Query Language) که در واقع شما می‌تونید با استفاده از اون با دیتابیس‌های رابطه‌ای کار بکنید؛ یعنی توش جدول جدید بسازید، داخلش جستجو کنید، ردیف بسازید، پاک کنید یا ویرایش کنید. نیاز هست که هر بک‌اند دولوپری این زبان رو بلد باشه و توانایی داشته باشه که کوئری‌های بهینه بنویسه (کوئری‌هایی که دیتابیس رو کُند نکنن).

ابزارهای شیءگرایی (ORM): همچنین نیازه که به یک لایبرری یا ORM(Object-Relational Mapping) که توی زبان برنامه‌نویسی شما وجود داره، مسلط بشید. ORMها به شما اجازه می‌دن که با مدل‌های شیءگرای زبان خودتون (مثلاً یک کلاس Python) با دیتابیس صحبت کنید، بدون اینکه مجبور باشید همه‌جا کوئری SQL خام بنویسید. این کار هم کُد شما رو تمیزتر می‌کنه و هم از لحاظ امنیتی خیلی بهتره.

نکته‌ای برای آینده دیتابیس‌ها: البته ما دیتابیس‌های دیگه‌ای هم داریم که بهشون می‌گیم NoSQL. این دیتابیس‌ها ساختار رابطه‌ای ندارن و برای انواع خاصی از داده‌ها (مثل لاگ‌ها یا داده‌هایی که ساختار متغیری دارن) بهتر عمل می‌کنن. ما در مقالات بعدی، مفصل در مورد NoSQL و انواع مختلف اون (مثل MongoDB) صحبت می‌کنیم و می‌گیم که کِی باید سراغ این مدل‌ها بریم. فعلاً تمرکزمون روی یادگیری عمقی دیتابیس‌های رابطه‌ای باشه.

نقشه‌های جا به جایی: مهاجرت‌ها(Migrations)

(تغییرات دیتابیس و نگهداری ساختار کد)

نرم‌افزارها عموماً تغییرات زیادی دارن. یعنی با گذر زمان، قابلیت‌های جدید اضافه و قدیمی‌ها ویرایش می‌شن. این تغییرات شامل تیبل‌های دیتابیسی شما هم هستن. مثلاً ممکنه امروز نیاز باشه یه ستون جدید به جدول کاربرها اضافه کنیم.

اینجاست که بحث Migrations(مهاجرت‌ها) وسط می‌آد. مهاجرت‌ها سازوکارهایی هستن که به شما اجازه می‌دن تغییرات ساختاری دیتابیس رو (مثل اضافه کردن یک ستون یا تغییر نام جدول) به‌صورت کدی و قابل ردیابی مدیریت کنید. این کار تضمین می‌کنه که دیتابیس شما توی محیط توسعه، تست و نهایی، همیشه ساختار یکسانی داشته باشه و بتونید به‌راحتی تغییرات رو اعمال کنید و در صورت نیاز برگردونید. این مفهوم به شدت برای کار تیمی و استقرار در سرور مهمه.

برای مایگرشن ابزارهای متفاوتی وجود که معمولاً به صورت پیش‌فرض روی همون فریم‌ورکی که باهاش کار می‌کنید وجود دارن مثلاً Django یا Sequelize توی Node.js. اما لایبرری‌ها مستقلی هم وجود دارن مثل Alembic توی پایتون یا Flyway/Liquibase که برای مدیریت این تغییرات استفاده میشن.

ابزارهایِ سپاهِ نرم‌افزار (ابزارهای ضروری)

(ابزارهای ضروری برای توسعه‌دهنده)

علاوه بر یادگیری مفاهیمی که دقیقاً مرتبط با نوشتن یک کد بک‌اند هست، شما باید یک‌سری ابزار رو هم بلد باشید که بتونید با استفاده از اون‌ها، کد خودتون رو توسعه بدید، نگهداری کنید و به بهترین شکل ممکن به سرور بفرستید.

الف. گیت(Git) برای کنترل نسخه

یکی از مهم‌ترین ابزارهایی که تقریباً باید همه برنامه‌نویس‌ها بلد باشن و بتونن باهاش کار کنن، یک ورژن کنترل هست که معروف‌ترین اون‌ها Git هست. گیت مثل یه ماشین زمان برای کد شماست.

باید بدونید مفاهیمی مثل Commit (ثبت یه بسته تغییرات)، Branch (انشعاب دادن برای کار روی یه ویژگی جدید)، Merge Request یا Pull Request (درخواست ادغام کد) و Merge (ادغام نهایی) چیه و چطوری باید با گیت کار کنید. تسلط بر گیت، همکاری شما رو توی تیم‌های نرم‌افزاری منظم و حرفه‌ای می‌کنه و اجازه نمی‌ده که کد یک نفر، کار بقیه رو خراب کنه. استفاده از پلتفرم‌هایی مثل GitHub و GitLab که برای مدیریت مخازن گیت استفاده می‌شن، توی این مرحله خیلی مهمه.

ب. داکر(Docker) برای ایزوله‌سازی محیط

بعد از اون، یکی از ابزارهایی که یک بک‌اند دولوپر مدرن باید بدونه، داکر(Docker) هست. احتمالاً شما هم این جمله رو شنیدید که برنامه‌نویس‌ها میگن: «روی سیستم من کار می‌کرد!»

داکر اومده که این مشکل رو حل کنه. داکر به شما یه محیط ایزوله می‌ده که نرم‌افزار شما با تمام وابستگی‌ها، کتابخونه‌ها و محیط عملیاتی که نیاز داره، داخل اونجا اجرا بشه. این یعنی شما مطمئن می‌شید که کدی که روی لپ‌تاپ خودتون می‌نویسید، دقیقاً همون‌طوری که انتظار می‌ره، روی سرور یا سیستم همکارتون هم کار می‌کنه.

باید بدونید که Image(تصویر) داکر چیه و چطوری از روی اون Container (کانتینر) ساخته می‌شه و این چطوری به ما کمک می‌کنه که کدمون رو توی هر محیطی، بدون دردسر زیاد استقرار(Deploy) بدیم.

نکته‌ای برای آینده: در آینده‌ای نزدیک، وقتی پروژه‌های ما بزرگ‌تر بشن، ممکنه مجبور بشیم ده‌ها یا صدها داکر کانتینر رو همزمان مدیریت کنیم. اینجاست که ابزارهایی مثل Kubernetes برای هماهنگی کانتینرها وارد میدون می‌شن، که بحث اونا رو هم توی قسمت‌های پیشرفته‌تر دنبال خواهیم کرد.

ج. تست و مستندسازی API

شما وقتی کد بک‌اند رو می‌نویسید، هنوز فرانت‌اندی وجود نداره که از طریق اون دکمه‌ها رو بزنید و ببینید کدتون کار می‌کنه یا نه. پس چطور باید تست کنید؟

اینجاست که ابزارهایی مثل Postman (یا جایگزین‌هایی مثل Insomnia) وارد می‌شن. این ابزارها نقش «فرانت‌اند مجازی» رو بازی می‌کنن و به شما اجازه می‌دن درخواست‌های HTTP (مثل Login کردن) رو به سرور بفرستید و جواب رو ببینید.

و ابزار مهم دیگه Swagger (یا OpenAPI) هست. وقتی شما یه API می‌سازید، برنامه‌نویس فرانت‌اند از کجا باید بدونه چه آدرسی رو صدا بزنه و چه دیتایی بفرسته؟ شما نباید این‌ها رو شفاهاً توضیح بدید. ابزارهایی مثل Swagger به صورت خودکار یک صفحه مستندات(Documentation) تمیز از روی کد شما می‌سازن که همه چیز رو توضیح میده. این یعنی زبان مشترک و مستند بین شما و تیم فرانت.