Laporan Pemrograman Mobile

Nama : Sebastian Abe Santoso

NIM : 2410817210002

Foundations

Deskripsi

09/02/2026 : Saya selesai menonton seluruh Modul 1 (bagian 1 - 6) dan mencatat bagian 3 - 6 (materi), di sini saya berfokus untuk membuat fondasi untuk pengetahuan, yang nantinya akan dicerna ulang dengan membanding catatan dengan apa yang di bahas di kursusnya.

10/02/2026 : Membaca ulang catatan, tetapi belum terlalu paham.

11/02/2026 : Saya pindah catatan ke GitHub agar mudah diperbaiki, serta catatan dapat diakses kapan saja. Link berada di https://github.com/SebastianAbeSantoso/Me/blob/main/LAPORAN_PEMROGRAMAN_MOBILE/CATATAN_MODUL1_ID.md

12/02/2026 : Membuat contoh laporan di Miro dengan seluruh catatan, yang nanti akan dipindah ke GitHub.

14/02/2026 : Menonton ulang serta mencatat bagian 1-2 (pembukaan kursus) untuk melengkapi seluruh catatan, serta memperbaiki format catatan.

15/02/2026 : Mencerna ulang materi untuk membuat insight & Impact, translasi seluruh catatan yang awalnya EN ke ID, dan memperbaiki ulang laporan & catatan.

Insight

Yang saya peroleh adalah gambaran umum tentang arsitektur perangkat lunak, arsitektur frontend, desain perangkat lunak, dan peran seorang frontend architect, detail lebih lanjut di bawah ini.

Seperti yang saya catat sebelumnya, arsitektur perangkat lunak dan desain dapat dipertukarkan karena keduanya merupakan bentuk keputusan, namun keduanya memiliki perbedaan yang jelas. Arsitektur perangkat lunak merupakan fondasi, artinya bersifat tingkat tinggi, sangat penting, dan keputusan yang harus diambil dengan benar sejak awal, sementara desain perangkat lunak lebih bersifat tingkat rendah, berfokus pada kode atau masalah desain, yaitu blok bangunan di atas fondasi. Karena keduanya tumpang tindih, hal ini dapat dilihat sebagai spektrum, sehingga menyatakan bahwa suatu masalah adalah arsitektur atau desain merupakan dikotomi palsu. Cara umum untuk menentukan di mana suatu masalah berada dalam spektrum adalah dengan mengajukan beberapa pertanyaan, seperti “apakah memerlukan pertimbangan dari tim-tim lain?”, “apakah sulit untuk diubah?”, “apakah akan memiliki dampak jangka panjang?” Semakin banyak pertanyaan yang terjawab, semakin architectural posisinya dalam spektrum.

Sama seperti arsitektur perangkat lunak, arsitektur frontend juga merupakan kumpulan keputusan penting untuk mempromosikan atribut kualitas dan properti lainnya, tetapi berada pada lapisan atau potongan vertikal sistem frontend.

Arsitektur perangkat lunak umumnya dibagi menjadi empat dimensi (Architecture style, Architecture characteristics, Architecture decisions, dan Logical components), tetapi hal ini bukanlah aturan baku seperti SOLID dalam OOP, karena pada aplikasi dan desain modern, batas antara keempat dimensi tersebut terus kabur. Keempat dimensi ini sangat penting dalam menggambarkan suatu arsitektur, karena semuanya bersatu menjadi satu kesatuan untuk sebuah aplikasi. Style atau structure menentukan bentuk sistem (Microservice, Monolitik, Event-driven, dll). Characteristics menggambarkan cara sistem beroperasi, biasanya diklasifikasikan dengan akhiran "-ility" atau "-ilities" (misalnya Agilities, Scalability). Decisions adalah batasan atau aturan yang spesifik dan dapat ditegakkan bagi pengembang untuk menyelaraskan pandangan mereka tentang sistem, pada dasarnya merupakan batas yang biasanya berbentuk pedoman. Akhirnya, Logical components adalah modul, seperti blok kode yang menjalankan serangkaian fungsi tertentu, mengelompokkan tugas serupa ke dalam partisi, pada dasarnya semua blok bangunan yang membentuk arsitektur atau perangkat lunak.

Terakhir, peran seorang architect frontend cukup samar, bahkan beberapa perusahaan mungkin tidak memiliki peran ini, tetapi dalam konteks ini, peran tersebut memiliki tiga ciri umum, yaitu menetapkan arah teknis, menerapkan pemikiran arsitektural, dan melakukan “glue work”. Secara sederhana, "glue work" menggambarkan tugas-tugas yang tidak terlihat/tidak dapat diukur yang diperlukan agar suatu proyek berhasil, dan sebagian besar bersifat non-teknis, seperti berkomunikasi, mendokumentasikan, onboarding, dan sebagainya. Poin penting yang perlu ditekankan adalah tidak perlu menjadi architect untuk peduli pada arsitektur. Meskipun mungkin memerlukan wewenang untuk membuat keputusan arsitektural, peduli pada arsitektur adalah tugas semua orang, bukan tanggung jawab eksklusif architect.

Dengan pengetahuan baru ini, saya mencoba brainstorming dan membayangkan bagaimana seorang frontend architect bekerja dan menciptakan arsitekturnya, dan menemukan kesamaan dengan peran game designer dalam game dev, karena keduanya memiliki bidang kerja yang serupa. Keduanya menyelaraskan pandangan tim tentang proyek (pemikiran sistem), menyeimbangkan imajinasi dan kenyataan, dan mengelola tim. Yang paling penting, keduanya adalah peran yang melakukan “pekerjaan penghubung”, seperti berkomunikasi antar peran, menetapkan deadline, membuat dokumen, dan sebagainya.

Impact

Topik ini benar-benar mengubah pandangan saya. Peran yang begitu penting ini ternyata jarang ditemukan di banyak perusahaan, namun kemudian saya mengetahui bahwa architect sering dianggap sebagai orang yang berada di "ivory tower", membuat diagram kompleks yang sulit diimplementasikan, sehingga peran ini dianggap “mati”. Di era modern, tanggung jawab ini didistribusikan di antara peran-peran senior+. Di perusahaan startup, peran spesifik ini sangat jarang (sering digabungkan dengan peran lain), tetapi di perusahaan besar, peran “glue work” yang spesifik ini wajib ada, meskipun tidak disebut “Frontend Architect”. Di Google, peran ini didasarkan pada manajemen teknis, bukan manajemen personel, disebut Individual Contributor, seperti L6 “Staff Software Engineer”, L7 “Senior Staff Engineer”, dan L8 “Principal Engineer”.

Memahami hal ini mengubah persepsi saya tentang tim yang biasanya memiliki manajer, tetapi perusahaan modern lebih memilih peran yang berbasis pada pekerjaan teknis (apa yang perlu dilakukan) daripada manajemen personel (siapa yang perlu melakukannya). Manajemen personel biasanya ditangani oleh manajer, yang lebih fokus pada manajemen orang daripada pekerjaan mereka.

https://progressivecoder.substack.com/p/is-the-software-architect-job-role Dalam newsletter ini, mereka memang menyebutkan bahwa “ivory tower architect” adalah alasan utama mengapa peran ini tidak disukai, yaitu orang-orang yang melakukan pekerjaan yang sama namun mendapatkan gelar khusus, serta mereka

yang membuat diagram yang rumit secara tidak perlu. Mereka juga menyebutkan bahwa arsitektur seharusnya menjadi perhatian setiap pengembang, bukan hanya architect, dan memperlakukan architect sebagai panduan, bukan manajer tunggal. Dengan alur kerja modern yang lebih kolaboratif, peran "solo architect" telah meredup, digantikan oleh architect yang kolaboratif.
