Tes Teknis Programmer Senior

JMC Indonesia

Oleh :

Aldi Pradana

(202501161)

TIM PRODUKSI

PT JIWA MUDA CIPTA INDONESIA

2026

A. Code Quality & Review

1. **Jelaskan kriteria kode yang dianggap readable dan maintainable pada tim besar.**

   Dimulai dari penamaan folder, fungsi, variable yang jelas sesuai dengan tujuan pembuatan fitur, tidak perlu di singkat2. Lalu menghindari penulisan kode yang berulang, buat class atau fungsi jika ada kode yang memang diperlukan di banyak tempat. Selanjutnya gunakan dotblock untuk menjelaskan suatu fungsi yang kompleks.

2. **Sebutkan 5 kesalahan umum kode junior dan bagaimana cara Anda mereviewnya secara konstruktif.**

   - variable sering di singkat2 : contoh review, tolong untuk variabel sebaiknya dibuat lebih deskriptif supaya mudah dipahami di masa depan. contoh : $ds -> $dataSPK
   - menulis variable tanpa mendefinisikan terlebih dulu : tolong sebaiknya diberi default value agar lebih aman dan menghindari error
   - penulisan kode yang berulang : kode ini dipakai di lebih dari 1 tempat, tolong dipisah dan dibuatkan fungsi helper untuk itu, biar rapi dan dan mudah debuggingnya.
   - validasi input data kurang informatif : tolong ditambahkan validasi input yang lebih jelas, supaya data yang masuk sesuai dan lebih gampang dicek kalau nanti ada error.
   - n+1 query : sabaiknya untuk query di dalam looping dihapus saja dan coba ganti menggunakan join, karna bisa itu bisa buat load data lambat.

3. **Bagaimana pendekatan Anda saat menemukan kode berjalan tapi desainnya buruk?**

   saya akan bertanya dulu ke SA, apakah memang sesuai rancangan UIUX? dan apakah boleh saya rapikan?. jika diperbolehkan saya akan merapikan desainnya agar terlihat rapi.

4. **Apa perbedaan code review pada junior vs middle developer?**

   Code review junior : mengecek apakah kode sudah benar, penamaan variabel jelas, dan gaya penulisan konsisten. Jika ada kesalahan, diberikan feedback spesifik beserta contoh perbaikan.

   Code review middle : melakukan review dasar seperti junior, ditambah menilai efisiensi, struktur, dokumentasi, dan kemudahan maintenance di masa depan.

5. **Kapan refactor wajib dilakukan, dan kapan harus ditunda?**

   Refactor wajib dilakukan jika fitur yang diminta sering error / outputnya tidak sesuai. Refactor harus ditunda ketika fitur akan segera digunakan user / sudah mendekati deadline, misal ada error perbaikan difokuskan secukupnya agar fungsi berjalan dengan benar tanpa mengubah struktur besar kode.

6. **Apa yang salah atau tidak ideal pada script-script berikut?**

   a.
```php
public function a(Request $r){
    $u = DB::table('users')->where('id',$r->i)->first();
    if($u){
        if($u->s==1){
            DB::table('o')->insert([
                'u'=>$u->id,
                't'=>date('Y-m-d'),
                's'=>0
            ]);
            return 1;
        }else{
            return 0;
        }
    }
    return -1;
}
```

   - penamaan fungsi, variable, nama table database, kolom table database dan return response fungsi harusnya lebih deskriptif.
   - belum ada validasi request.
   - pengecekan if ada 2 bisa diringkas jadi 1 if saja, misal s tipe datanya boolean, maka bisa dituliskan if ($u?->s === true)

- gunakan eloquent yang lebih ringkas jika hanya memunculkan 1 data dan menyimpan data yaitu find dan insert , contoh laravel misal table u menggunakan model User dan s menggunakan model Order $user = User::find($r->i), Order::insert([...]);

b.

```
function process($x){
    if($x){
        if($x->a){
            if($x->b){
                if($x->c){
                    // do something
                }
            }
        }
    }
}
```

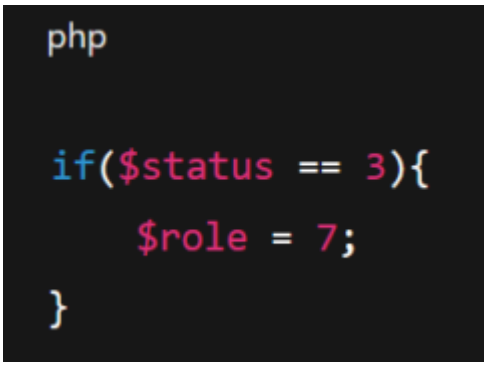
- nama properti tidak deskriptif.
- terlalu banyak if bersarang.
- lebih baik dijadikan 1 level saja pengkondisiannya dan tiap kondisi yang gagal langsung diberikan response.

c.

```
SELECT * FROM a,b,c
WHERE a.id=b.aid
AND b.cid=c.id
AND a.s=1
AND c.t='X'
AND (b.q>0 OR b.q IS NULL);
```

- nama table dan kolom tidak deskriptif
- sebaiknya select * diganti dengan memilih kolom spesifik yang diperlukan saja.
- gunakan join agar query mudah dibaca

d.



- value $status sebaiknya bukan angka 1,2,3.., gunakan yang lebih deskriptif, misal dibuat global constant, misal **public const STATUS_PENDING = 3;public const STATUS_SUCCESS = 4;**
- sebaiknya menggunakan strict kondisi (===).
- variable $role perlu didefinisikan terlebih dahulu agar tidak terjadi error.

B. Version Control & Conflict

1. **Jelaskan penyebab umum terjadinya conflict saat commit/push**

   Conflict terjadi ketika 2/lebih orang mengubah bagian yang sama dari sebuah file pada commit yang berbeda.

2. **Uraikan langkah sistematis menangani conflict tanpa merusak history.**
   - jika terjadi konflik akan ada warning rejected, contohnya saat melakukan git push ke branch
   - lalu git pull
   - di terminal akan memunculkan info file mana yang konflik
   - lalu akan muncul di file yang conflik tambahan kode, seperti <<<<<<< hingga HEAD (bagian ini berisi kode perubahan kita) dan kode tambahan setelah ======= hingga >>>>>>> adalah perubahan dari user lain, pilih atau sesuaikan mana yang dibutuhkan lalu hapus block tambahan tsb
   - save file, git add . lalu commit lagi dengan git commit -m “fix konflik …”
   - terakhir git push

3. **Bagaimana kebijakan branching yang ideal untuk tim dengan 5–10 developer?**

   menurut saya, nanti akan terdapat 3 jenis branch, main, test, branch-developer-task. Setiap developer yang akan melakukan modifikasi kode, perlu membuat branch dengan nama branch sesuai task yang akan dibuat bisa dengan format nama - nama tasknya apa. Tujuannya untuk meminimalkan terjadinya konflik.

4. **Kapan rebase lebih tepat dibanding merge, dan risikonya?**

   rebase lebih tepat ketika project hanya di develop hanya 1 orang, jika penggunaan git rebase dengan banyak tim / branch akan menimbulkan masalah saat sinkronisasi pada branch yang digunakan bersama, karena terdapat perubahan id commit ketika melakukan rebase.

C. Task Breakdown, Estimasi & Taiga

1. **Breakdown task dan estimasi waktu yang realistis dari masing-masing task**

No	Task	Estimasi (Jam)
1	Perancangan database	2
2	Setup Laravel dan Theme Admin 5	3
3	Modul Login	3
4	Modul Kelola User	3
5	Dashboard	1
6	Modul Data Pegawai	4
7	Setting Tunjangan Transport	1
8	Modul Tunjangan Transport	4
9	Modul Log	2
10	Modul Presensi - Rekap	2
11	Modul Presensi - Import	4

D. Dokumentasi & Komunikasi Teknis

1. **Jenis dokumentasi apa yang wajib dibuat oleh programmer senior?**
    - cara install aplikasi
    - struktur database
    - alur aplikasi
    - dokumentasi api (jika ada)
    - aturan penulisan
2. **Bagaimana Anda membaca dan memvalidasi dokumentasi teknis yang ambigu?**

    perlu saya trial bagian teknis yang ambigu itu dan mengajak diskusi dengan pembuat dokumentasi teknis sebelumnya.

3. **Jelaskan peran dokumentasi dalam mencegah technical debt**

    membantu mempercepat penanganan pengembangan fitur / perbaikan bug di masa mendatang dan juga membantu mengkonsistenkan kode dimasa mendatang agar mudah di pahami.