BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan & Organisasi

Pelaksaan kerja magang di Agile Technica beralamat di Universitas Multimedia Nusantara, New Media Tower Lv.11, Ruang Parseus, Curug Sangereng, Kec. Klp. Dua, Tangerang, Banten. Proses kerja magang berlangsung selama 5 bulan kerja sebagai *Software Developer Internship* di bawah pengawasan *co-founder* sekaligus *executive* perusahaan.

3.2 Tugas yang dilakukan

Tugas yang diberikan selama proses kerja magang adalah pembuatan REST API pada *web application*. API merupakan sebuah *software* yang memungkinkan aplikasi untuk saling terhubung antara satu dengan yang lain. Dalam hal ini, REST API digunakan sebagai alat komunikasi data dari *back-end* ke *front-end*. Data yang diproses merupakan data jadwal perjanjian pasien dengan dokter yang didapat dari resepsionis klinik.

Proses pembuatan API menggunakan bahasa pemrogaman Java. Penggunaan MongoDB sebagai *database* dikarenakan lebih fleksibel dalam penggunaannya. Penyimpanan data pada MongoDB menggunakan struktur JSON, sehingga MongoDB mempunyai nilai fleksibilitas. Dilihat dari segi kompleksitas data yang akan diolah, maka penggunaan MongoDB merupakan pilihan yang sesuai.

Dalam pembuatan *web application*, *code editor* yang digunakan adalah IntelliJ IDEA. IntelliJ IDEA sendiri merupakan *code editor* yang di dalamnya sudah terintegrasi oleh **git** (*sub-version*) tanpa perlu memerlukan aplikasi pihak ketiga.

3.3 Uraian Pelaksanaan Kerja Magang

3.3.1 Proses Pelaksanaan

Nama Kegiatan	Minggu												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Pengenalan Bahasa Pemrograman Java dan API													
Pengenalan Project													
Pengembangan Aplikasi													

Tabel 3.1 Tabel Timeline Kerja Magang

Minggu pertama dan kedua dilakukan proses instalasi dan *setup account* di beberapa aplikasi yang akan dipakai dalam melakukan pekerjaan. Aplikasi yang dibutuhkan, yaitu Slack, Owncloud, Gitlab, Intellij IDEA, Postman, MongoDB, dan Taiga. Selain instalasi, dilakukan juga pembelajaran terhadap bahasa pemrograman Java dengan melakukan riset secara *on-line* dan memahami *code* di *project* yang sedang dijalankan. Selain bahasa pemrograman, dipelajari juga struktur *file* pada *project* sebelum terlibat secara langsung ke dalam *project*.

Setelah mempelajari arah *project* yang dilakukan, penulis terlibat secara langsung dalam pengembangan *web application,* terutama di bidang *back-end*. *Team Leader* telah menyiapkan *repository*, sehingga penulis bisa melakukan *clone repository* dan melakukan pengembangan.

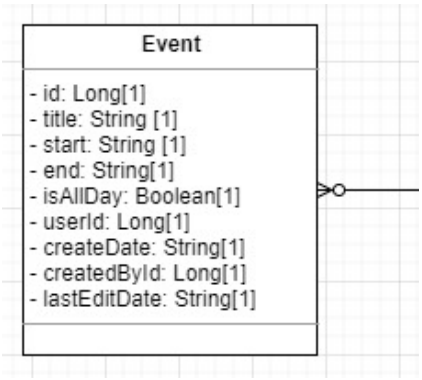
Setelah *clone repository*, penulis mulai mengerjakan tugas sesuai yang diberikan. Setelah *sprint* berakhir, *resepsionis* mulai mengikuti *sprint planning, retrospective*, dan *demo app* kepada *client*. Penulis mengikuti *sprint planning* yang merupakan bagian dari metode *agile* agar dapat menilai kesulitan *task* yang

dikerjakan oleh tim, *retrospective* untuk mengetahui kinerja *team*, dan *demo app* untuk memberikan demo aplikasi kepada *client* tentang *progress* yang dikerjakan oleh *team*.

3.3.2 Uraian Kerja Magang

Uraian pengerjaan kerja magang meliputi *Class Diagram*, *Activity Diagram* dan implemetasi aplikasi. *Class Diagram* digunakan sebagai landasan struktur aplikasi yang digunakan pada sistem. *Activity Diagram* sebagai instruksi *developer* dalam mengerjakan aplikasi. Implementasi aplikasi sebagai gambaran hasil kerja *developer* dari sisi *resepsionis* saat menjalankan aplikasi.

A. Class Diagram



Gambar 3.1 Class Diagram Event

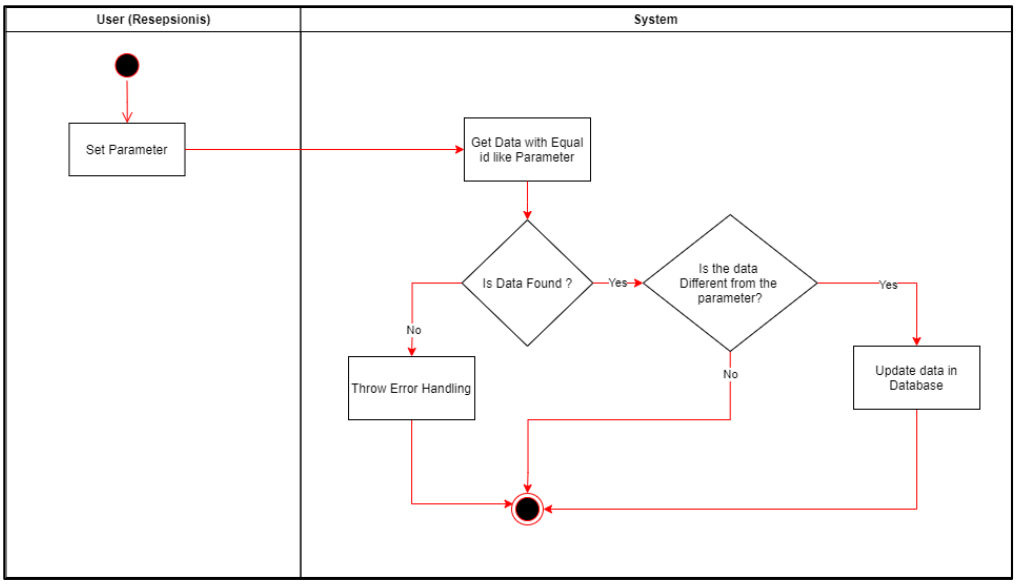
Gambar 3.1 merupakan struktur yang digunakan saat membuat class. Class ini diberi nama Event sesuai kegunaannya yaitu menyimpan data dari sebuah Event. Class ini terdiri dari variabel:

- id : Nama dari sebuah json
- title : Judul atau Tujuan janji
- start : Awal dari jam janji

- end : Akhir dari jam janji
- isAllDay : Menentukan janji akan dilakukan satu hari penuh
- resepsionisId : Id dari dokter
- createDate : Hari dibuatnya janji
- createdById : Id dari pembuat janji
- lastEditDate : Hari diubah janji bila terjadi perubahan

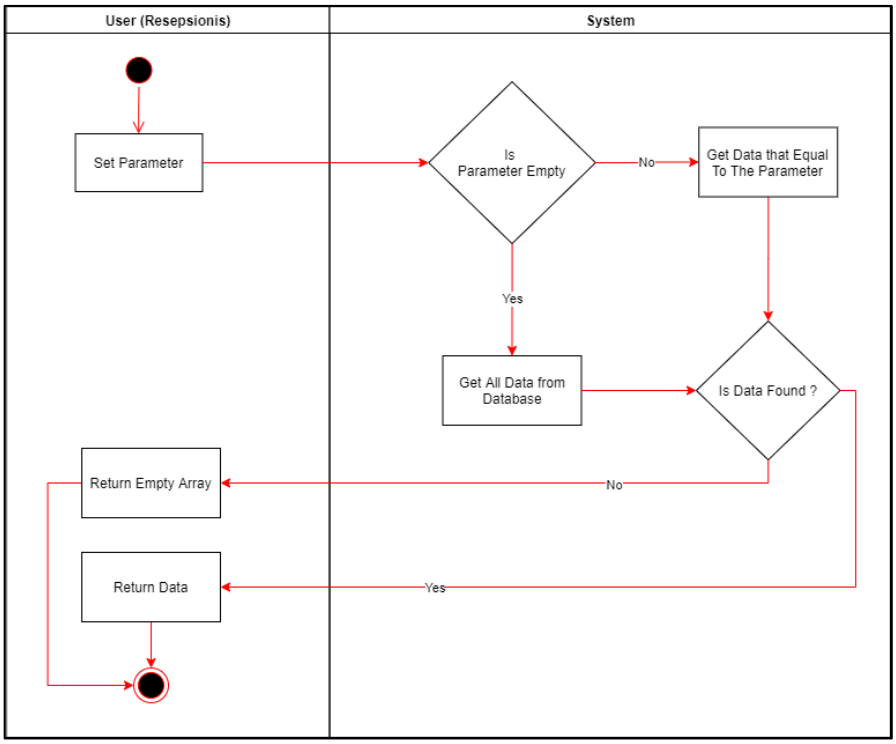
B. Activity Diagram

Activity Diagram akan menjelaskan fungsi masing-masing Rest API. *Activity Diagram* yang disajikan adalah aktivitas berjalannya sebuah API dari *input* sampai *output* yang diberikan serta data-data didalamnya. API yang dikerjakan oleh penulis adalah Post, Get, Update, dan Delete.



Gambar 3.2 Activity Diagram API Post Event

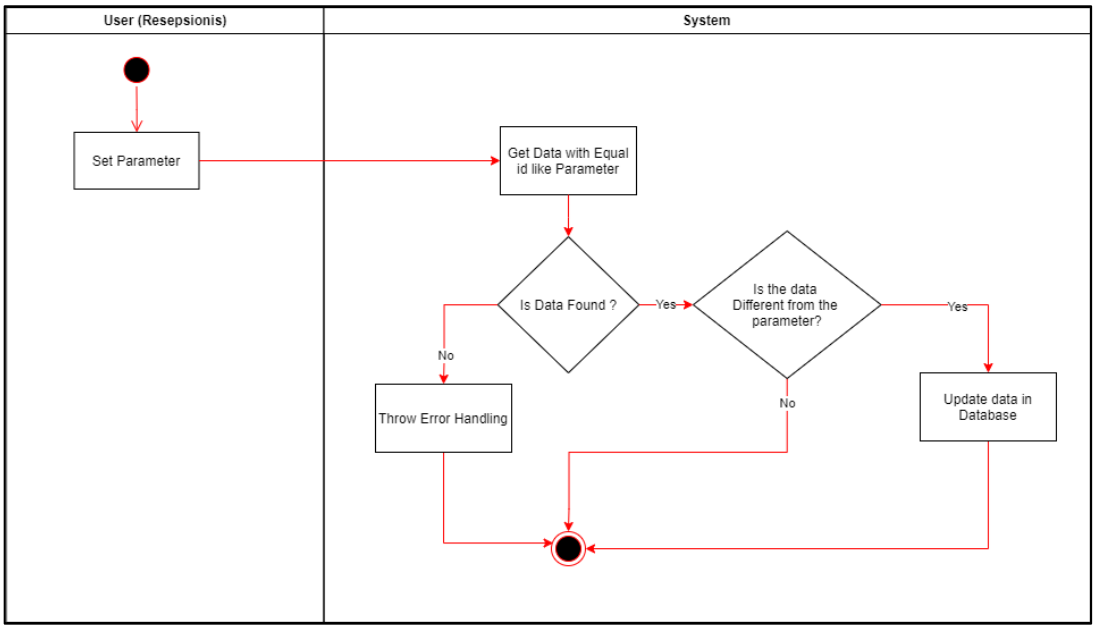
Pada API Post Event, API akan menyimpan tanggal yang telah disepakati oleh pasien untuk diisi dengan waktu kunjungan dengan dokter yang dibantu oleh resepsionis. Informasi yang telah didapat dari pasien akan dimasukkan oleh resepsionis dari *front-end*. Setelah data dimasukkan, API akan melakukan pengecekan di *database*. Bila data sudah pernah dimasukkan, id Event akan melanjutkan dari data sebelumnya. Bila tidak, akan dibuat mulai dari angka awal. Setelah itu, API akan memasukkannya ke dalam *database*.



Gambar 3.3 Activity Diagram API Get Event

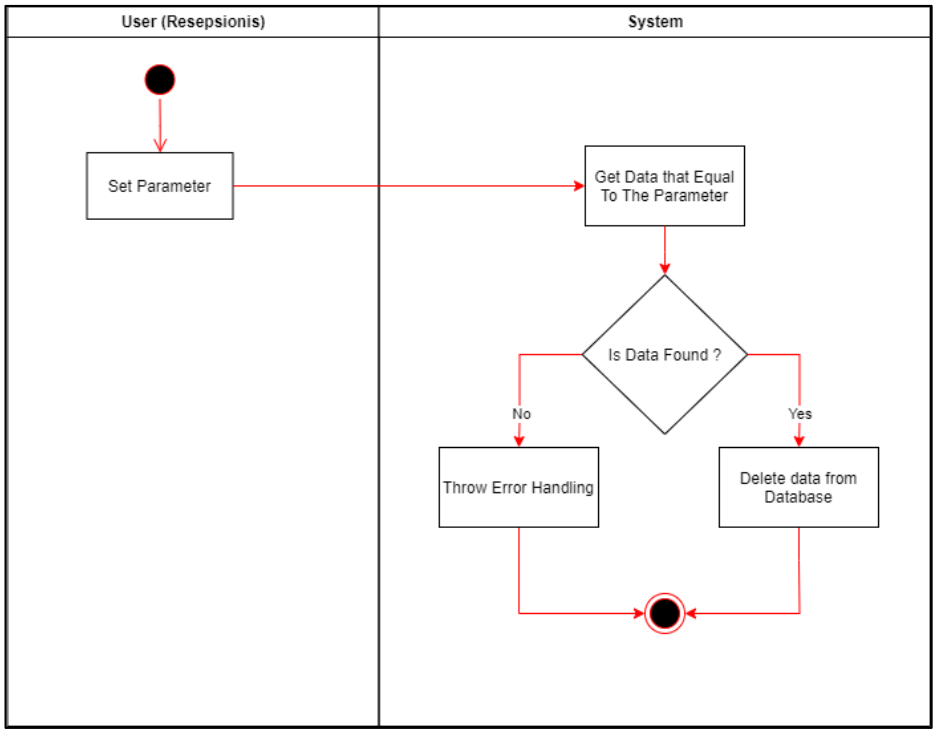
Pada API Get Event, API akan memberikan *output* Event yang tersimpan pada *database*. Ada dua pilihan saat menggunakan API ini, resepsionis bisa mendapatkan semua Event dengan tidak mengirimkan parameter ke API dan juga dengan mengirimkan parameter ke API untuk mendapatkan Event sesuai dengan parameternya.

Pada tampilan *front-end* akan disediakan nama dokter sebagai parameter. Setelah parameter dimasukkan, API akan melakukan pengecekan pada *database*. Bila ada data yang sesuai dengan parameter, *back-end* akan memberikan *output* data yang memiliki parameter. Apabila tidak ada data yang memiliki parameter yang sesuai, *back-end* akan mengirimkan *array* kosong yang menandakan data tidak ditemukan.



Gambar 3.4 Activity Diagram API Update Event

Pada API Update Event, API akan memperbarui Event yang sebelumnya sudah tersimpan di *database*. API Update Event digunakan apabila terjadi perubahan pada data yang sudah dibuat sebelumnya. Seperti contoh, pasien ingin merubah tanggal yang telah ditentukan. API akan menerima *input* parameter sesuai dengan API Post dengan tambahan id Event. API akan melakukan perbandingan dari parameter dengan data yang ada pada *database*. Bila ada data yang berbeda pada database, maka data akan diperbarui sesuai yang dimasukkan oleh resepsionis. Setelah diperbarui, tanggal pada lastEditDate akan diperbarui juga sebagai penanda bahwa data telah diubah.

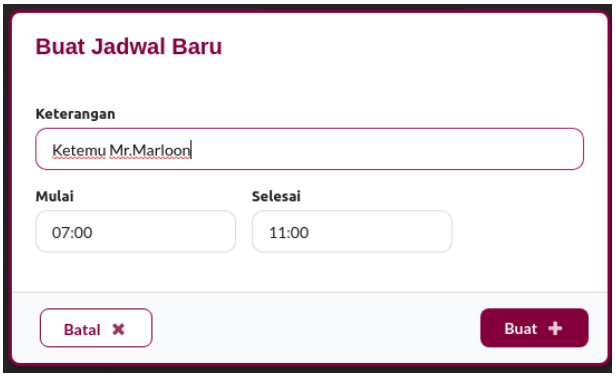


Gambar 3.5 Activity Diagram API Delete Event

Pada API Delete Event, API akan menghapus Event yang sebelumnya sudah pernah dibuat. API Delete Event digunakan apabila dari pihak dokter ataupun pasien ada kendala untuk hadir pada hari tersebut. *Front-end* akan memberikan parameter id Event, dan bila ada kecocokan, data akan dihapus dari database. Jika data tidak ditemukan, API akan memberikan *output error handling* yang menandakan data tidak ditemukan.

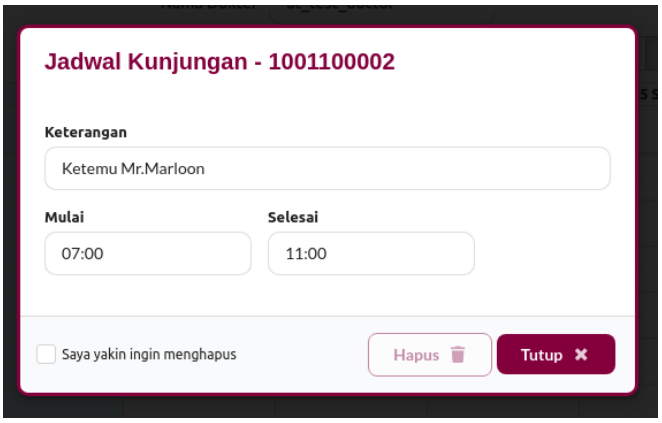
C. Implementasi

Berikut implementasi API terhadap UI yang sudah dirancang dan diuji.



Gambar 3.6 Tampilan Pembuatan Event

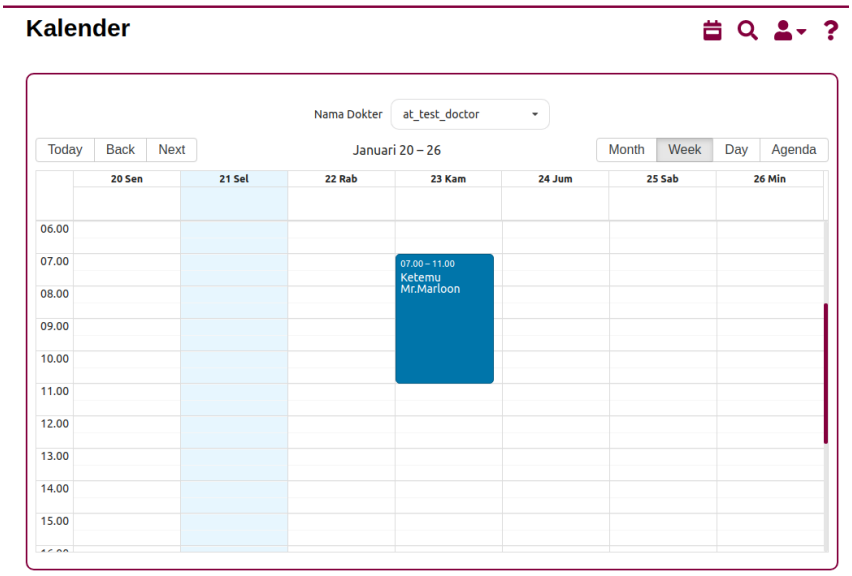
Pada Gambar 3.6, judul berita di bagian keterangan dan jam pertemuan pasien dengan dokter akan dimasukkan oleh resepsionis. Jika keterangan dan jam sudah benar, API Post Event akan dijalankan dengan menekan tombol buat.



Gambar 3.7 Tampilan Event

Pada Gambar 3.7, ketika resepsionis menekan jadwal akan muncul *pop up* yang menampilkan informasi mengenai jadwal yang telah dipilih. API Delete Event akan dijalankan ketika menekan tombol hapus. Apabila ada informasi yang diubah

(keterangan atau jam), API Update Event akan dijalankan ketika menekan tombol tutup.



Gambar 3.8 Tampilan Seluruh Kalender

Pada Gambar 3.8 merupakan tampilan awal kalender yang menampilkan seluruh jadwal sesuai dengan dokter yang dipilih. API Get Event akan dijalankan ketika resepsionis membuka halaman kalender.

3.3.3 Kendala yang Ditemukan

Pada pelaksanaan kerja magang, kendala yang dihadapi adalah sebagai berikut:

- Tidak memiliki pengetahuan sebelumnya tentang pembuatan API dan cara kerja MongoDB dan Git.
- Adanya *miscommunication* dalam *team,* sehingga terjadi perubahan pada API di dalam pengembangan.
- *Requirement client* yang selalu berubah karena terlalu banyak *client* yang ikut bertanggungjawab.

3.3.4 Solusi Atas Kendala yang Ditemukan

Kendala yang telah dijelaskan sebelumnya terdapat solusi yang dapat melancarkan jalannya magang dan membuat penulis berkembang. Solusi untuk menghadapi kendala tersebut adalah sebagai berikut:

- Melakukan *research* terlebih dahulu dari internet dan juga berdiskusi dengan *team leader* mengenai suatu masalah dan menyelesaikannya bersama.
- Melakukan konfirmasi satu sama lain mengenai *task* yang dikerjakan tentang requirement yang diperlukan di *front-end*.
- Berdiskusi lebih lanjut antar *team* untuk hasil terbaik dan terefisien untuk aplikasi berdasarkan saran dari *client*.