

PENGEMBANGAN *BACKEND* PADA STARTUP SAJILOKA



Disusun Oleh:

N a m a : Rayhan Mahardhika W.

NIM : 18523055

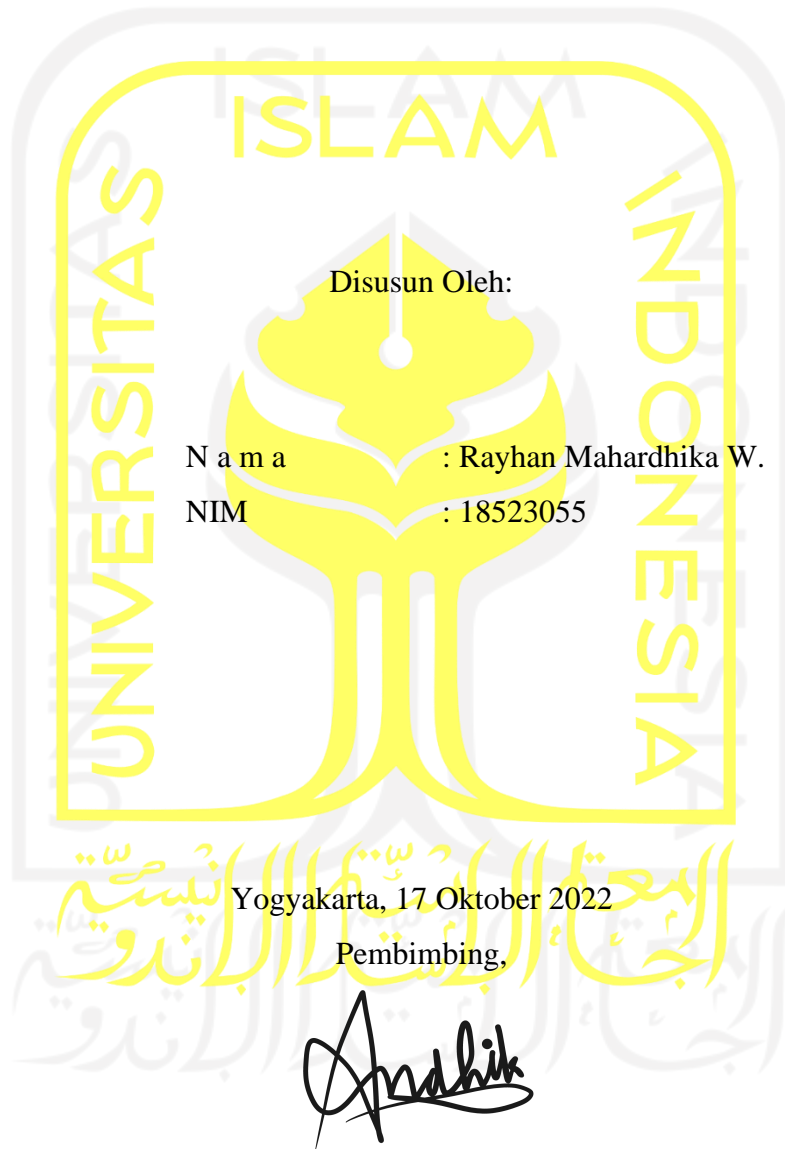
**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN *BACKEND* PADA STARTUP SAJILOKA

TUGAS AKHIR



Andhik Budi Cahyono, S.T., M.T.

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN *BACKEND* PADA STARTUP SAJILOKA

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 20 Oktober 2022

Tim Penguji

Andhik Budi Cahyono, S.T., M.T.



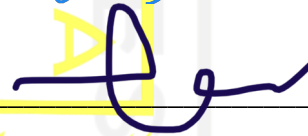
Anggota 1

Dr. Syarif Hidayat, S.Kom., M.I.T.



Anggota 2

Erika Ramadhani, S.T., M.Eng.



Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia



Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Rayhan Mahardhika Wijaya

NIM : 18523055

Tugas akhir dengan judul:

PENGEMBANGAN *BACKEND* PADA STARTUP SAJILOKA

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

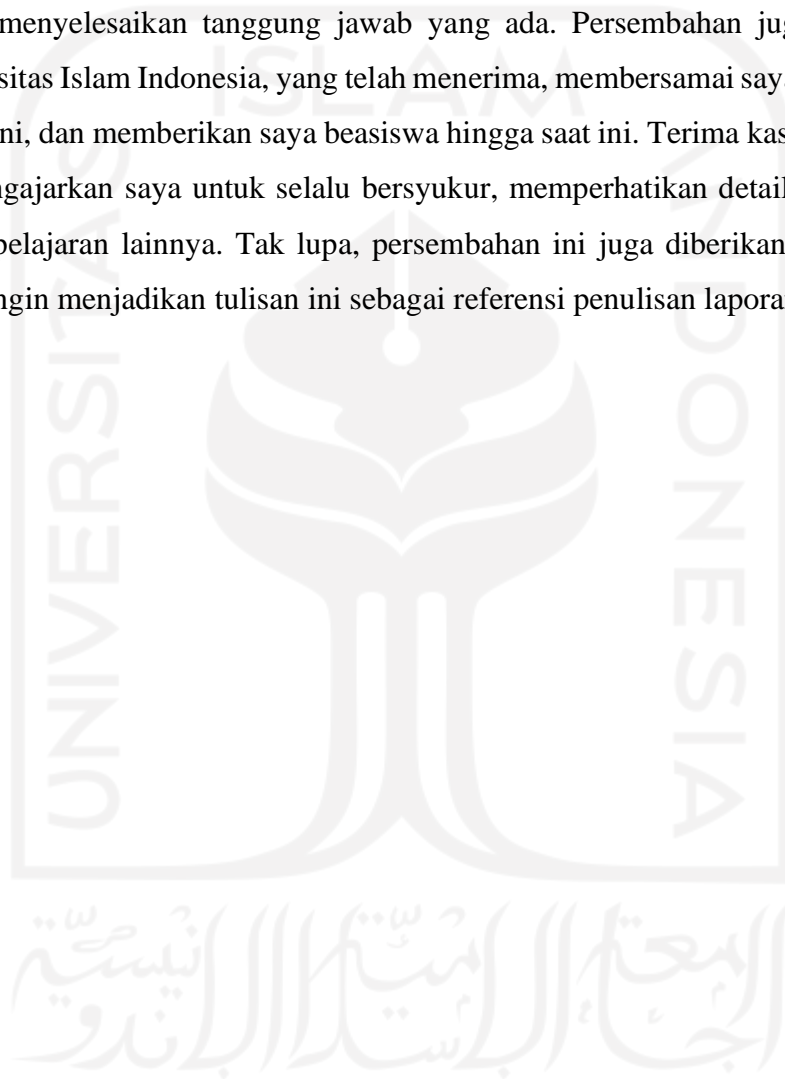
Yogyakarta, 17 Oktober 2022



Rayhan Mahardhika Wijaya

HALAMAN PERSEMBAHAN

Puji syukur atas rahmat Allah Yang Maha Kuasa, yang telah memberikan saya kesempatan untuk menyelesaikan laporan tugas akhir. Laporan ini merupakan persembahan istimewa yang saya berikan untuk siapapun yang telah mendukung saya, baik dalam bentuk materil maupun moril. Persembahan besar saya berikan kepada orang tua, sahabat, dan teman-teman saya yang telah menjadi support system selama ini, sehingga saya bisa berkembang sejauh ini dan menyelesaikan tanggung jawab yang ada. Persembahan juga saya berikan terhadap Universitas Islam Indonesia, yang telah menerima, membersamai saya tumbuh hingga bisa belajar di sini, dan memberikan saya beasiswa hingga saat ini. Terima kasih kepada setiap orang yang mengajarkan saya untuk selalu bersyukur, memperhatikan detail, berpikir kritis, serta pelajaran-pelajaran lainnya. Tak lupa, persembahan ini juga diberikan kepada seluruh pembaca yang ingin menjadikan tulisan ini sebagai referensi penulisan laporan tugas akhir.



HALAMAN MOTO

“Great companies are built on great products.”

– Elon Musk

“The biggest risk is not taking any risk... In a world that changing quickly, the only strategy that is guaranteed to fail is not taking risks.”

- Mark Zuckerberg



KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh...

Alhamdulillah, segala puji syukur atas kehadiran Allah SAW yang telah memberikan rahmat, hidayah, serta inayahnya, sehingga penulisan Laporan Tugas Akhir yang berjudul “PENGEMBANGAN *BACKEND* PADA STARTUP SAJILOKA” dapat diselesaikan tepat waktu. Shalawat serta salam senantiasa tercurahkan kepada Rasul dan kekasih-Nya, Muhammad SAW, yang telah membawa kita dari jaman jahiliah ke jaman yang terang benderang penuh dengan ilmu.

Adapun latar belakang dari selesainya Laporan Tugas Akhir ini ialah sebagai persyaratan Tugas Akhir Jalur Perintisan Bisnis dan selesai nya studi yang dilakukan di Program Studi Informatika - Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Dalam menjalani seluruh aktivitas, baik magang dan penyusunan Laporan Tugas Akhir, terdapat banyak pihak yang terlibat. Penulis ingin menyampaikan rasa terima kasih kepada seluruh pihak yang terlibat dalam penulisan ini. Beberapa di antaranya adalah sebagai berikut:

1. Kedua orang tua saya, yang selalu memberikan semangat, dukungan, dan memahami penulis selama ini.
2. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Dthomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Andhik Budi Cahyono, S.T., M.T., selaku dosen pembimbing yang mengayomi dan membimbing penulis selama magang hingga penyusunan laporan dapat diselesaikan.
5. Salsabila Zahirah Pranida, yang telah memberikan semangat, nasihat, dan dukungan selama perkuliahan dilakukan.
6. Seluruh tim Sajiloka, khususnya Arief Rahman, yang telah bersedia bersama-sama mengembangkan aplikasi Sajiloka.

Atas bantuan berbagai pihak, penulis akhirnya dapat menyelesaikan laporan tugas akhir dengan sebaik mungkin. Namun, penulis menyadari bahwa laporan tugas akhir yang dibuat masih jauh dari kata sempurna, sehingga penulis sangat membutuhkan kritik dan saran untuk membuat laporan tugas akhir menjadi lebih baik. Akhir kata, semoga laporan ini dapat bermanfaat bagi kita semua.

Wassalamu'alaikum Warrahmatullahi Wabarakatuh...

Yogyakarta, 15 Juli 2022



(Rayhan Mahardhika Wijaya)



SARI

Aplikasi Sajiloka merupakan sebuah aplikasi yang dikembangkan pada kegiatan perintisan bisnis yang diadakan oleh 1000 Startup Digital. Pada pengembangan aplikasi Sajiloka, penulis berperan sebagai *hacker* yang bertugas mengembangkan aplikasi dari sisi *backend*. Aplikasi *backend* dikembangkan untuk mendukung kebutuhan transaksi dari aplikasi sisi *frontend*. Terdapat tiga tahapan yang dilakukan dalam pengembangannya, yakni inisiasi proyek, eksekusi proyek, dan evaluasi proyek. Eksekusi proyek dilakukan dengan menggunakan *Sprint*. Sedangkan, pada evaluasi nya menggunakan pengujian perangkat lunak *unit testing*. Pengembangan yang dilakukan menghasilkan aplikasi *backend* yang bisa dijalankan melalui aplikasi Postman serta telah ter-*deploy* pada layanan hosting gratis Heroku untuk memudahkan proses pengembangan. Hasil dari evaluasi yang telah dilakukan, didapatkan bahwa seluruh unit pengujian mendapatkan nilai *pass*, yang mana dapat diartikan bahwa sistem bekerja dengan baik.

Kata kunci: *Backend, Scrum, Startup, Unit Test*

GLOSARIUM

Data Mock	merupakan <i>dummy data</i> yang digunakan dalam pengujian <i>unit test</i> pada <i>layer repository</i> yang merefleksikan data input dan output
SQL	suatu jenis basis data yang bersifat relational antar tabel
API	Singkatan dari Application Programming Interface yang memberikan dukungan kepada aplikasi sisi <i>front end</i> .
Go	bahasa pemrograman yang digunakan dalam pengembangan Web API
Aplikasi <i>Backend</i>	merupakan aplikasi yang memberikan dukungan data kepada aplikasi front end serta tidak dapat dilihat langsung oleh pengguna
Aplikasi <i>Frontend</i>	merupakan aplikasi yang menampilkan tampilan dari sebuah perangkat lunak langsung kepada pengguna
NoSQL	suatu jenis basis data yang bersifat dokumen dan tidak relational antar tabel/collection
MVP	singkatan dari <i>Minimum Viable Product</i> yang berarti sebuah produk siap pakai dengan fitur yang minimal
ClickUp	aplikasi manajemen proyek yang digunakan dalam kegiatan Scrum

DAFTAR ISI

HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR.....	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO.....	vi
KATA PENGANTAR.....	vii
SARI.....	ix
GLOSARIUM	x
DAFTAR ISI	xi
DAFTAR TABEL	xiii
DAFTAR GAMBAR.....	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.1 Batasan Masalah	3
1.2 Tujuan Pengembangan	3
1.3 Manfaat Pengembangan	3
1.4 Sistematika Penulisan	3
BAB II LANDASAN TEORI	5
2.1 <i>Startup</i>	5
2.1.1 <i>Hustler</i>	5
2.1.2 <i>Hipster (Designer)</i>	6
2.1.3 <i>Hacker</i>	6
2.2 Gerakan 1000 Startup Digital	6
2.2.1 Tahap <i>Ignition</i>	7
2.2.2 Tahap <i>Networking</i>	7
2.2.3 Tahap <i>Workshop</i>	8
2.2.4 Tahap <i>Hacksprint</i>	8
2.2.5 Tahap <i>Bootcamp</i>	8
2.2.6 Tahap <i>Hatch</i> (Inkubasi)	9
2.3 <i>Scrum</i>	9
2.3.1 Kegiatan Scrum	9
2.3.2 Artefak Scrum	11
2.4 Pengembangan Aplikasi <i>Backend</i>	12
2.5 REST API	13
2.5.1 Prinsip Desain REST API	14
2.5.2 Cara Kerja REST API	15
2.6 Pengujian Perangkat Lunak	16
BAB III METODE PENGEMBANGAN	18
3.1 Inisiasi Proyek	18
3.2 Eksekusi Proyek	19
3.2.1 <i>Sprint 1</i>	19
3.2.2 <i>Sprint 2</i>	22
3.2.3 <i>Sprint 3</i>	24
3.3 Evaluasi Proyek	25
3.3.1 <i>Auth Service</i>	26
3.3.2 <i>Menu Service</i>	27
3.3.3 <i>Ingredient Service</i>	29

3.3.4	<i>Category Service</i>	30
3.3.5	<i>Occasion Service</i>	31
3.3.6	<i>Difficulty Service</i>	32
3.3.7	<i>Menu Service (Admin)</i>	33
	BAB IV HASIL DAN REFLEKSI	35
4.1	Hasil Pengembangan Aplikasi	35
4.2	Hasil Pengujian Aplikasi	44
4.2.1	Sisi Pengguna (Consumer)	44
4.2.2	Sisi Pengelola (Admin)	45
4.3	Refleksi	46
4.3.1	Kendala dan Hambatan	46
4.3.2	Tantangan	47
4.3.3	Capaian	47
4.3.4	Wawasan Kegiatan	47
	BAB V KESIMPULAN DAN SARAN	49
5.1	Kesimpulan	49
5.2	Saran	49
	DAFTAR PUSTAKA	xv
	LAMPIRAN	xvi



DAFTAR TABEL

Tabel 4.1 Hasil akhir <i>unit testing</i> Web API Pengguna	45
Tabel 4.2 Hasil akhir <i>unit testing</i> Web API Admin.....	45



DAFTAR GAMBAR

Gambar 2.1 <i>Startup Golden Triangle</i>	5
Gambar 2.2 <i>Scrum Events</i>	9
Gambar 2.3 Alur ReST API.....	14
Gambar 2.4 <i>Unit testing</i> pada GO.....	17
Gambar 3.1 Alur Metode Pengembangan.....	18
Gambar 3.2 <i>Task Sprint 1</i>	20
Gambar 3.3 Desain basis data <i>Sprint 1</i>	21
Gambar 3.4 <i>Task Sprint 2</i>	23
Gambar 3.5 Desain basis data <i>Sprint 2</i>	23
Gambar 3.6 <i>Task Sprint 3</i>	25
Gambar 3.7 <i>Unit testing</i> fungsi <i>GenerateToken (Auth Service)</i>	26
Gambar 3.8 <i>Unit testing</i> fungsi <i>ValidateToken (Auth Service)</i>	27
Gambar 3.9 <i>Unit testing</i> fungsi <i>GetAllMenu (Menu Service)</i>	27
Gambar 3.10 <i>Unit testing</i> fungsi <i>GetMenuById (Menu Service)</i>	28
Gambar 3.11 <i>Unit testing</i> fungsi <i>GetProcessingStepsById (Menu Service)</i>	29
Gambar 3.12 <i>Unit testing</i> fungsi <i>GetMenuIngredients (Ingredient Service)</i>	30
Gambar 3.13 <i>Unit testing</i> fungsi <i>GetAllCategories (Category Service)</i>	30
Gambar 3.14 <i>Unit testing</i> fungsi <i>GetAllByMenuId (Category Service)</i>	31
Gambar 3.15 <i>Unit testing</i> fungsi <i>GetAllOccasions (Occasion Service)</i>	31
Gambar 3.16 <i>Unit testing</i> fungsi <i>GetAllByMenuId (Occasion Service)</i>	32
Gambar 3.17 <i>Unit testing</i> fungsi <i>GetAllDifficulties (Difficulty Service)</i>	33
Gambar 3.18 <i>Unit testing</i> fungsi <i>GetById (Difficulty Service)</i>	33
Gambar 3.19 <i>Unit testing</i> fungsi <i>CreateMenu (Menu Service)</i>	34
Gambar 4.1 API Masuk Pengguna.....	36
Gambar 4.2 API Daftar Pengguna	37
Gambar 4.3 API Dapatkan Akun Pengguna	37
Gambar 4.4 API Daftar <i>Category</i>	38
Gambar 4.5 API Daftar <i>Occasion</i>	38
Gambar 4.6 API Daftar Produk	39
Gambar 4.7 API Detail Produk By ID	40
Gambar 4.8 API Bahan dan Cara Memasak Produk.....	41
Gambar 4.9 API Daftar Produk (Admin).....	41

Gambar 4.10 API Detail Produk (Admin)	42
Gambar 4.11 API Daftar <i>Category</i> (Admin)	43
Gambar 4.12 API Daftar <i>Difficulty</i> (Admin)	43
Gambar 4.13 API Simpan Data Produk (Admin)	44



BAB I PENDAHULUAN

1.1 Latar Belakang

Kegiatan 1000 Startup Digital atau dapat disebut dengan “1000SD” merupakan sebuah ajang perintisan bisnis digital yang membantu masyarakat penggiat *startup* untuk dapat menyalurkan ide, serta berbondong-bondong mengimplementasikan ide tersebut ke dalam bentuk produk digital siap pakai. Kegiatan 1000SD dilaksanakan pada bulan Oktober 2021. Terdapat beberapa tahapan kegiatan yang perlu dilalui peserta yaitu *ignition*, *networking*, *workshop*, *hacksprint*, *bootcamp*, dan *hatch*. Kegiatan pengembangan ide serta aplikasi dilaksanakan mulai dari tahap *hacksprint* hingga *bootcamp* setelah itu akan dilanjutkan dengan penyisihan untuk melanjutkan ke tahap inkubasi atau *hatch*.

Ide yang diusulkan merupakan aplikasi bernama Sajiloka yang menyediakan kebutuhan memasak sehari-hari melalui paket menu masakan siap saji, yang mana dapat disesuaikan dengan selera pengguna. Sajiloka muncul sebagai ide bisnis karena memiliki peluang pasar dengan permasalahan yang sudah dibuktikan atau divalidasi pada kegiatan *hacksprint*. Setelah ide Sajiloka selesai dikembangkan, maka ide akan dilanjutkan ke dalam tahap *bootcamp*. Tahap *bootcamp* merupakan tahapan saat ide akan diimplementasi menjadi sebuah aplikasi digital tahap MVP (*Most Viable Product*).

Mekanisme pengembangan aplikasi MVP Sajiloka ini menggunakan dua aplikasi. Dua aplikasi terpisah tersebut yaitu *frontend* dan *backend*. Melalui penjelasan yang diberikan oleh (Singhal, 2021), aplikasi pada sisi *frontend* merupakan aplikasi yang memberikan tampilan aplikasi yang interaktif kepada pengguna, serta biasanya disebut *client-side*. Sedangkan, aplikasi pada sisi *backend* memiliki tanggung jawab untuk memastikan sisi *frontend* bekerja dengan lancar dengan cara memberikan dukungan data yang sesuai dengan kebutuhan yang diperlukan pada sisi *frontend*. Aplikasi pada sisi *backend* juga berperan untuk melakukan komunikasi antara sistem yang berbeda, serta tempat untuk mengimplementasikan modul saintifik, seperti *machine learning*, *deep learning*, dan lain sebagainya. Aplikasi pada sisi *backend* biasanya disebut juga dengan *server-side*.

Parameter yang menjadi acuan dalam kegiatan 1000SD adalah pengembangan aplikasi yang telah sampai pada tahap MVP pada saat fase *bootcamp*. Tahap MVP pada fase *bootcamp* sudah dapat menangani transaksi oleh pengguna (*client*). Hal tersebut merupakan tantangan

yang diterima oleh tim pengembang, karena tim hanya terdiri dari dua orang dengan pengalaman yang terbatas.

Aplikasi sisi *backend* yang dikembangkan berupa *Application Programming Interface* (API) yang menggunakan standar arsitektur ReST (*Representational State Transfer*). Aplikasi juga dibangun menggunakan bahasa pemrograman Golang (Go) dengan menggunakan basis data MySQL. Penggunaan bahasa pemrograman Go dalam pengembangan aplikasi pada sisi *backend* Sajiloka dilatarbelakangi oleh beberapa faktor, di antaranya adalah keunggulan dan kemudahan. Bahasa pemrograman Go memiliki keunggulan dibandingkan dengan bahasa pemrograman lainnya. Kemudian, bahasa pemrograman Go merupakan bahasa pemrograman yang cukup familiar dengan tim pengembang *backend*.

Keunggulan lain mengenai bahasa pemrograman Go juga telah dipaparkan oleh (Bai, n.d.), yang mana bahasa pemrograman tersebut merupakan bahasa yang cepat dan ringan dibandingkan dengan bahasa pemrograman lainnya untuk pengembangan *backend*, seperti bahasa pemrograman Java dan Python. Hal ini disebabkan karena bahasa pemrograman Go merupakan bahasa yang sudah *ter-compile*, sehingga dapat langsung dipahami oleh *processor*. Bahasa pemrograman Go juga mudah dipelajari oleh para pengembang yang sebelumnya sudah pernah mempelajari dan menggunakan bahasa pemrograman C ataupun Java. Kemudahan ini diakibatkan karena bahasa pemrograman Go memiliki pendekatan yang sama dengan bahasa pemrograman C dan Java, sehingga para pengembang tidak membutuhkan waktu lama untuk mempelajari bahasa pemrograman Go. Skalabilitas yang dimiliki oleh bahasa pemrograman Go juga lebih baik dibandingkan bahasa pemrograman lain, seperti Java. Bahasa pemrograman Go juga lebih hemat *memory*, sehingga bahasa pemrograman Go mampu menjalankan pemrograman yang lebih besar dan berat dengan lebih baik daripada Java.

Penggunaan basis data SQL pada aplikasi sisi *backend* didasari dan didukung oleh penjelasan dari (GeeksforGeeks, 2020), yang mana SQL dinilai lebih familiar dan mudah digunakan, yang mana disebabkan dari banyaknya dokumentasi yang telah tersebar luas di dunia maya. SQL juga lebih cocok digunakan untuk menyimpan data transaksional dan relasional, yang mana sesuai dengan kebutuhan yang diperlukan oleh aplikasi Sajiloka. Tidak menutup kemungkinan, apabila di masa yang akan datang, aplikasi Sajiloka menggunakan basis data SQL dan NoSQL bersamaan. Hal ini dikarenakan terdapatnya data-data yang cocok dan lebih efisien apabila digunakan basis data NoSQL, seperti data riwayat.

Penjelasan lebih lanjut terkait hasil dari kegiatan pengembangan akan dijelaskan di dalam laporan. Melalui hasil dari kegiatan pengembangan aplikasi Sajiloka pada sisi *backend*,

diharapkan mampu mendukung kebutuhan dan transaksi data yang dibutuhkan pada sisi *frontend*.

1.1 Batasan Masalah

Guna memperjelas fokus penulisan, batasan masalah perlu ditetapkan. Berikut merupakan batasan-batasan masalah dalam pengembangan aplikasi Sajiloka sisi *backend*:

- a. Laporan ini melaporkan semua kegiatan yang telah dilaksanakan pada kegiatan perintisan bisnis khusus nya pengembangan aplikasi Sajiloka sisi *backend* sehingga hasil akhir dari laporan ini adalah sebuah refleksi dari kegiatan pengembangan yang telah dilakukan.
- b. Semua *task* yang dikerjakan sudah didefinisikan sesuai dengan desain yang diberikan oleh *hipster* sebelum *sprint* berlangsung.
- c. Hasil aplikasi *backend* yang dikembangkan berupa kumpulan API yang akan digunakan oleh aplikasi *frontend* dan pada laporan ini akan divisualisasikan melalui aplikasi Postman.
- d. Pengujian hasil aplikasi *backend* menggunakan *unit testing* dengan bertujuan untuk memastikan aplikasi sudah berjalan dengan baik.

1.2 Tujuan Pengembangan

Tujuan dari pengembangan yang dilakukan terhadap aplikasi Sajiloka pada sisi *backend* (*server-side*) adalah guna mendukung transaksi data yang dibutuhkan oleh sisi *frontend* (*client-side*).

1.3 Manfaat Pengembangan

Pengembangan aplikasi Sajiloka sisi *backend* diharapkan mampu memberikan dukungan data kepada aplikasi sisi *frontend*, di antaranya sebagai berikut:

- a. Mampu memberikan data yang sesuai dan dibutuhkan oleh sisi *frontend*
- b. Mampu mengikuti kebutuhan data dari sisi *frontend* yang berubah-ubah dan berkembang
- c. Mampu menjaga agar sisi *frontend* berjalan dengan baik

1.4 Sistematika Penulisan

Sistematika penulisan bertujuan untuk memudahkan pemahaman isi dan tujuan dari laporan tugas akhir. Berikut merupakan sistematika yang terdiri dari lima bab yaitu:

- a. **Pendahuluan**

Bab ini membahas tentang latar belakang pengembangan aplikasi, batasan masalah, tujuan pengembangan, manfaat pengembangan, dan sistematika penulisan terkait pengembangan aplikasi Sajiloka pada sisi *backend*.

b. Landasan Teori

Bab ini membahas tentang teori-teori yang berhubungan dengan pengembangan aplikasi sebagai landasan pengembangan aplikasi Sajiloka sisi *backend*.

c. Metode Pengembangan

Bab ini membahas secara seksama kegiatan pengembangan aplikasi Sajiloka sisi *backend* pada setiap iterasi *sprint* serta kegiatan pengujian aplikasi.

d. Hasil dan Refleksi

Bab ini membahas tentang hasil dari pengembangan aplikasi Sajiloka sisi *backend*, hasil pengujian menggunakan *unit test*, dan refleksi dari kegiatan pengembangan aplikasi yang telah dilaksanakan.

e. Kesimpulan dan Saran

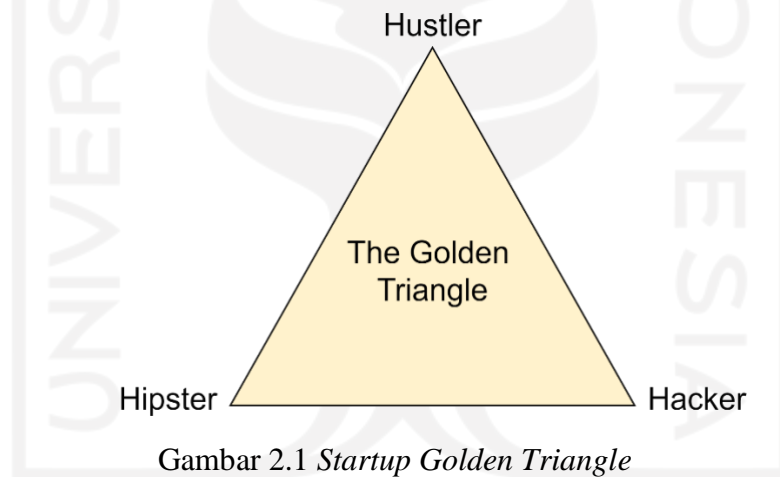
Bab ini membahas tentang kesimpulan dan saran yang memberikan rangkuman terhadap semua hasil dari kegiatan pengembangan aplikasi Sajiloka sisi *backend*.

BAB II

LANDASAN TEORI

2.1 *Startup*

Startup atau perusahaan rintisan, khususnya pada bidang perangkat lunak yang kemudian dikategorikan berdasarkan tantangan yang dihadapi seperti memiliki pengalaman pada pengembangan dan operasional yang sedikit atau tidak memiliki sama sekali, memiliki sumber daya yang terbatas sehingga memiliki fokus utama untuk memberikan produk kepada pelanggan secepatnya, menghadapi banyak tekanan dari berbagai pihak (klien, investor, partner, dan pesaing) yang mempengaruhi pengambilan keputusan dalam perusahaan, menghadapi dinamika teknologi dan pasar yang dihadapi sehingga mendorong perusahaan rintisan untuk penggunaan teknologi yang disruptif untuk dapat memasuki pasar yang berpotensi tinggi (Paternoster, Giardino, Unterkalmsteiner, Gorschek, & Abrahamsson, 2014).



Gambar 2.1 *Startup Golden Triangle*

Menurut (Thomas, 2014) dalam implementasi nya, untuk menginisiasi sebuah startup adalah dengan membentuk sebuah tim yang memiliki komposisi tim dijuluki The Golden Triangle, yaitu tiga peran utama yang terdiri dari *Hustler*, *Hipster*, dan *Hacker*.

2.1.1 **Hustler**

Peran ini berfokus kepada sisi pengembangan bisnis dari sebuah *startup*. Di samping penerapan inovasi teknologi yang mungkin menjadi salah satu kunci kesuksesan sebuah startup, pengembangan sebuah bisnis yang kuat juga tidak kalah penting sebagai landasan dalam startup untuk terjun ke pasar. Secara detail *hustler* akan fokus terhadap lima hal, yang

di antaranya adalah *customer development*, *product management*, *selling*, *blogging*, dan *recruiting*.

2.1.2 *Hipster (Designer)*

Peran ini berfokus kepada pendekatan visual dari sisi produk maupun pemasaran. Pada sisi pemasaran, *hipster* harus memiliki kemampuan untuk menciptakan sebuah citra yang menarik pada startup. Selain itu pada sisi produk, *hipster* harus mampu membuat semua kebutuhan-kebutuhan calon pengguna kedalam *User Experience* dan *User Interface (UI/UX)*. Secara detail hal-hal yang perlu difokuskan oleh *hipster* yaitu, *landing page*, *product design*, *design testing*, *branding*, dan *technical*.

2.1.3 *Hacker*

Peran ini memiliki tanggung jawab dalam implementasi teknologi yang digunakan dalam realisasi produk startup. *Hacker* harus merupakan orang yang berkecimpung dalam dunia pemrograman dan mengikuti perkembangan teknologi. Fokus utama *hacker* satu-satunya adalah realisasi produk.

2.2 Gerakan 1000 Startup Digital

Gerakan 1000 *Startup Digital* yang kemudian disingkat menjadi 1000SD atau #1000StartupDigital merupakan suatu program sebuah upaya bahu membahu untuk menggerakkan ekosistem startup digital di Indonesia untuk dapat saling terkoneksi, saling berbagi pengetahuan, dan pengalaman. Gerakan ini diinisiasi sejak tahun 2016, dan diharapkan dapat mendorong terciptanya dalam mencetak *startup* yang dapat menjadi solusi atas permasalahan dengan memanfaatkan teknologi digital (1000SD, n.d.). Gerakan 1000 *Startup Digital* hingga saat ini telah dilaksanakan di beberapa kota besar, seperti Medan, Toba, Pekanbaru, Batam, Pontianak, Balikpapan, Jakarta, Bandung, Semarang, Surakarta, Yogyakarta, Surabaya, Malang, Denpasar, Mataram, Makassar, Manado, Kupang, Ambon, dan Jayapura.

Fokus utama dalam Gerakan 1000 *Startup Digital* adalah untuk mendorong terwujudnya ekosistem *startup* digital yang kolaboratif dan inklusif, dengan visi menciptakan solusi bagi permasalahan bangsa dengan memanfaatkan teknologi digital. Sedangkan, tujuan dari adanya Gerakan 1000 *Startup Digital* adalah memberikan peluang kewirausahaan berbasis teknologi digital ke seluruh penjuru nusantara.

Pelaksanaan Gerakan 1000 *Startup* Digital memiliki enam tahap kegiatan yang perlu dilalui oleh peserta, di antaranya adalah *Ignition*, *Networking*, *Workshop*, *Hacksprint*, *Bootcamp*, dan *Hatch*. Sampai saat ini, sudah lebih dari sepuluh rintisan bisnis yang lolos dari tahap inkubasi dan berhasil mendirikan *startup*. Beberapa alumni yang telah dicetak oleh Gerakan 1000 *Startup* Digital antara lain adalah Agendakota, Amacall, Andil, Bantuternal, Bizhare, Botika, Garda Pangan, Invita, Kandang.in, Kinerja Bank, Lactashare, Lindungi Hutan, dan lain sebagainya. Alumni *startup* yang telah berhasil berasal dari wilayah kota yang berbeda-beda dan focus pada sektor yang beragam pula.

2.2.1 Tahap *Ignition*

Tahap *ignition* merupakan kegiatan inisiasi awal berupa seminar pemberian pengetahuan dasar seputar ekosistem *startup*, pola pikir perintis, dan permasalahan pada sektor yang terfokus. Seminar akan diberikan oleh pelaku bisnis dan regulator *startup* di Indonesia. Sesi seminar juga menghadirkan pendiri perusahaan rintisan untuk berbagi pengalamannya dalam membangun perusahaan.

Tahapan ini umumnya dilaksanakan secara *hybrid*, yakni parallel secara daring dan luring. Tahapan *ignition* juga terdiri atas tiga kegiatan utama, di antaranya adalah *tech conference*, *ecosystem networking*, serta *startup showcase and demo day*. Pada kegiatan *tech conference*, pada *early-founder* dan calon *startup founder* akan mengikuti kegiatan seminar dengan berbagai topik seputar gambaran, perkembangan, serta potensi dari *startup* digital di Indonesia saat ini.

Sementara, pada *ecosystem networking*, pada pelaku *startup* dapat menemukan peluang untuk terhubung dengan *stakeholder* yang potensial dan secara eksklusif. Terakhir, pada sesi *demo day*, sepuluh *early-founder* yang terpilih dari tahapan *hatch* yang diselenggarakan oleh Gerakan 1000 *Startup Digital* berkesempatan untuk mempresentasikan *business model* yang telah terkurasi di hadapan *stakeholder*.

2.2.2 Tahap *Networking*

Tahap *Networking* merupakan kegiatan berjejaring yang dilakukan secara daring untuk saling mengenal ide, sektor fokus pilihan, dan kemampuan yang dimiliki satu sama lain. Sesi berkenalan atau *networking* dilakukan antar peserta, yang kemudian dibagi menjadi enam sektor focus yang di antaranya adalah pertanian, kelautan, kesehatan, Pendidikan, pariwisata, dan logistik. Salah satu tujuan dalam tahapan ini adalah untuk dapat membentuk kelompok perusahaan rintisan, khususnya bagi yang belum memiliki kelompok.

2.2.3 Tahap *Workshop*

Kemudian, tahap *Workshop* merupakan suatu kegiatan belajar secara daring yang terbagi atas empat modul, yaitu *Hipster*, *Hacker*, *Hustler*, dan *General*. Sesi *workshop* atau yang dikenal juga sebagai pembekalan ini berfokus untuk menggali lebih dalam permasalahan yang dipilih dan memvalidasi target pelanggan menggunakan metode *design sprint* yang dilakukan selama dua hari.

2.2.4 Tahap *Hacksprint*

Selanjutnya, tahap *Hacksprint* merupakan tahap *sprint* yang membutuhkan waktu dua hari, yang di antaranya adalah *networking*, *prototyping*, dan *pitching*. Tahap *hacksprint* dikenal juga sebagai tahap pembuatan produk yang dilakukan secara cepat. Pembuatan produk secara cepat juga mengombinasikan metode *hackathon* dan *design sprint*.

Pada hari pertama diselenggarakannya tahap *hacksprint*, fokus pada kegiatan adalah melakukan *problem validation* dan *generating prototype to build* untuk memetakan permasalahan dan memvalidasi nya. Adapun beberapa materi yang perlu dikerjakan oleh peserta meliputi *mapping stakeholder*, *expert sharing*, penyusunan *how might we*, *heat mapping*, hingga *sprint goal setting*.

Kemudian, pada hari kedua penyelenggaraan akan berfokus pada *feedback*, *crazy eight*, *solution sketch*, dan *storyboard*. *Storyboard* yang perlu disusun oleh seluruh tim yang mengikuti tahapan ini adalah penyusunan *storyboard* mengenai *startup* yang dibuat, kemudian mempresentasikan hasilnya. Kegiatan ini kemudian akan menghasilkan keluaran berupa *solution sketch* sebagai bahan pertimbangan dalam penyusunan *Minimum Viable Product* (MVP).

2.2.5 Tahap *Bootcamp*

Tahap *Bootcamp* yang dilakukan setelah *hacksprint* adalah tahap sesi *mentoring* mendalam untuk menyiapkan strategi peluncuran produk. Tahapan ini merupakan sesi pendampingan yang dilakukan dengan konsep 1-on-1 dengan mentor pilihan. Pendampingan yang dilakukan akan berfokus pada produk, pemasaran, bisnis, dan perundingan ide yang dilakukan selama kurang lebih dua hingga tiga hari.

Para mentor yang akan mendampingi berasal dari berbagai *startup* yang berbeda dan ahli pada bidangnya masing-masing, seperti Philip Morris International, Tiket.com, Alpha JWC, Monshot Ventures, dan lain sebagainya. Beberapa skills yang dimiliki oleh para mentor adalah *skill analyst*, *brand design*, *branding*, *data*, *design thinking*, *investment*, *market research*,

quality assurance, hingga *UX writer*. Para mentor ini juga berasal dari dalam dan luar negeri, seperti Australia, Berlin, Brunei, Canada, London, Singapore, Thailand, dan Tokyo.

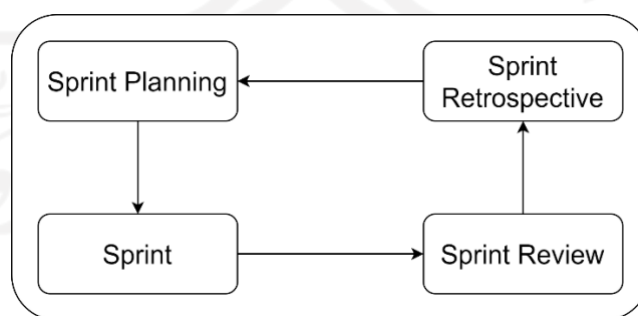
2.2.6 Tahap *Hatch* (Inkubasi)

Terakhir, tahapan *Hatch* atau inkubasi adalah sebuah program pendampingan yang tidak berbeda jauh dengan tahapan *bootcamp* sebelumnya. Namun, pada sesi ini, program pendampingan akan berfokus untuk meningkatkan satu metrik yang penting, yang menggambarkan *traction* dari rintisan bisnis yang dibangun dalam waktu kurang lebih enam minggu. Sama halnya seperti pada tahap *bootcamp*, pada tahap inkubasi, tim akan didampingi oleh mentor-mentor yang telah berpengalaman pada bidangnya. Hasil akhir yang diharapkan dari tahap ini adalah *startup* akan mencapai *product market-fit* melalui pendampingan mentor-mentor tersebut.

2.3 Scrum

Scrum adalah sebuah metode pengembangan berbentuk kerangka kerja (*framework*) yang merupakan bagian dari “*Agile Metodologies*”. Scrum memiliki sifat ringan dan adaptif, ringan berarti mampu diimplementasikan ke dalam berbagai kompleksitas proyek yang akan dikembangkan. Scrum secara singkat merupakan sebuah kerangka kerja yang membantu suatu tim atau organisasi untuk mencapai tujuan dalam pengerjaan proyek yang kompleks melalui pendekatan iteratif yang menghasilkan solusi adaptif.

2.3.1 Kegiatan Scrum



Gambar 2.2 *Scrum Events*

Tujuan utama kegiatan yang terdapat dalam Scrum adalah untuk melakukan inspeksi dan adaptasi pada artefak yang dibuat. Terdapat lima kegiatan yang ideal nya terjadi dalam satu

iterasi Scrum, di antaranya adalah *the sprint*, *sprint planning*, *daily scrum*, *sprint review*, dan *sprint retrospective*.

a. *The Sprint*

Sprint merupakan inti utama dari metode Scrum di mana ide-ide diubah menjadi sebuah solusi yang memiliki *value*. *Sprint* merupakan sebuah wadah di mana kegiatan Scrum lain dilaksanakan seperti *Sprint Planning*, *Daily Scrum*, *Sprint Review*, dan *Sprint Retrospective*. Durasi setiap *Sprint* selalu sama antara dua hingga empat minggu. Terdapat beberapa poin yang harus diperhatikan saat *Sprint* berlangsung yaitu tidak boleh ada perubahan yang mengancam *Sprint Goal*, tidak boleh menurunkan kualitas, *Product Backlog* disesuaikan secukupnya, cakupan *Sprint* dibahas dan diperjelas oleh *Product Owner* setiap ada temuan pada saat *Sprint* berlangsung.

b. *Sprint Planning*

Sebelum dimulainya *Sprint*, akan dilakukan perencanaan atau yang disebut *Sprint Planning* sebagai kegiatan yang menginisiasi *Sprint*. Menurut buku *Scrum Guide* terdapat beberapa topik yang biasanya dibahas saat melaksanakan *Sprint Planning* yaitu penetapan *Sprint Goal* yang memiliki nilai berharga bagi produk sehingga nilai keseluruhan produk bisa meningkat, memilih hal yang terdapat pada *Product Backlog* untuk dikerjakan pada *Sprint* berupa *Sprint Backlog*.

c. *Daily Scrum*

Menurut buku *Scrum Guide*, tujuan utama dari kegiatan *Daily Scrum* adalah untuk melakukan inspeksi terhadap proses menuju *Sprint Goal* dan adaptasi terhadap *Sprint Backlog* sesuai kebutuhan. Kegiatan ini dilaksanakan pada waktu yang sama setiap hari dengan rentang waktu berkisar 15 menit. *Developers* boleh melakukan cara atau Teknik apa saja untuk memberikan *delivery* terhadap *Sprint Goal*. Pada kegiatan ini *Developers* juga akan memberikan rencana untuk pengerjaan tugas pada hari berikutnya, hal ini secara langsung akan meningkatkan manajemen diri dari anggota tim.

d. *Sprint Review*

Tujuan dari kegiatan ini adalah untuk melakukan inspeksi terhadap hasil dari *Sprint* dan menentukan adaptasi di kemudian hari. Dalam kegiatan ini tim Scrum akan menunjukkan hasil kerja pada *Sprint* dan proses kearah *Product Goal* dari hasil yang telah dikerjakan. Dalam kegiatan inilah stakeholder di luar tim Scrum akan mengetahui apa yang telah dicapai dalam

Sprint dan pengaruh dalam lingkungan stakeholder tersebut. Ideal nya waktu yang diperlukan dalam melaksanakan kegiatan ini adalah empat jam dalam jangka waktu *Sprint* satu bulan, jika jangka waktu *Sprint* lebih pendek biasanya akan lebih pendek juga waktu kegiatan ini.

e. *Sprint Retrospective*

Tujuan utama dari kegiatan ini adalah untuk meningkatkan kualitas dan efektivitas dari kegiatan *Sprint* selanjutnya. Tim akan melakukan inspeksi terkait kegiatan *Sprint* yang telah dilaksanakan pada beberapa poin yaitu temuan yang berguna dalam *Sprint* yang sangat membantu, permasalahan yang dihadapi, dan pemecahan permasalahan yang dilakukan. Tim akan mengidentifikasi hal-hal paling penting yang dapat membantu meningkatkan efektivitas *Sprint* dan akan diterapkan dalam *Sprint* berikutnya. Ideal nya waktu yang diperlukan dalam melaksanakan kegiatan ini adalah tiga jam dalam jangka waktu *Sprint* satu bulan, jika jangka waktu *Sprint* lebih pendek biasanya akan lebih pendek juga waktu kegiatan ini.

2.3.2 Artefak Scrum

Artefak dalam Scrum merepresentasikan sebuah nilai atau hasil kerja yang dibuat selama Scrum dijalankan oleh tim atau organisasi, untuk mengakomodasi sebuah pengerjaan proyek. Artefak dalam Scrum dirancang setransparan mungkin sebagai informasi kunci dari inspeksi, sehingga semua orang memiliki dasar pemikiran yang sama terkait apa yang terjadi dan adaptasi apa yang harus dilakukan. Terdapat tiga artefak yang didefinisikan dalam Scrum, di antaranya adalah *product backlog*, *sprint backlog*, dan *increment*.

a. *Product Backlog*

Artefak ini merupakan sumber paling utama untuk pengerjaan yang akan dilakukan oleh Scrum Team. Dalam artefak ini terdapat istilah komitmen yang harus dirasakan semua anggota tim Scrum yaitu *Product Goal*. *Product Goal* menggambarkan sebuah kondisi produk yang menjadi acuan target untuk tim Scrum dalam membuat sebuah rencana. *Product Goal* merupakan tujuan jangka panjang tim Scrum, sehingga setiap tujuan harus diselesaikan atau dihilangkan untuk bisa menuju ke tujuan lain.

b. *Sprint Backlog*

Artefak ini terdiri dari tiga hal yaitu *Sprint Goal*, setiap butir *Product Backlog* yang telah dipilih untuk dilaksanakan dalam *Sprint*, dan sebuah rencana aksi untuk melakukan *delivery* untuk *Increment*. Artefak ini dibuat dan dipergunakan oleh *Developers*, visualisasi yang terdapat pada artefak ini sangat detail dan mencerminkan rencana yang teknis sehingga

merupakan acuan yang digunakan oleh *Developer* dalam mengerjakan tugas. Artefak ini dapat dilakukan adaptasi yang dibutuhkan selama *Sprint* berlangsung. Dalam *Sprint Backlog* terdapat istilah komitmen yang digunakan yaitu *Sprint Goal*. *Sprint Goal* merupakan sebuah komitmen yang diterapkan oleh *Developer*. *Sprint Goal* dibuat pada saat kegiatan *Sprint Planning* berlangsung dan akan ditambahkan ke dalam *Sprint Backlog*. Jika hasil dari pengerjaan tidak sesuai ekspektasi maka akan dilakukan negosiasi cakupan *Sprint Backlog* kepada *Product Owner* tanpa memberikan efek kepada *Sprint Goal*.

c. *Increment*

Artefak ini merupakan sebuah pijakan menuju *Product Goal*. Setiap penambahan *Increment* merupakan akumulasi pada keseluruhan *Increment*. Pada akhir kegiatan Scrum dan setelah dilakukan kegiatan *Sprint Review*, maka dilakukan *Increment*, yang merupakan tempat untuk mendokumentasi hasil dari *Sprint* yang telah dinyatakan selesai melalui *Definition of Done*. Berfokus kepada *Definition of Done* merupakan sebuah kondisi dari *Increment* yang sesuai dengan ekspektasi yang diperlukan oleh *Product*. Waktu di mana butir *Product Backlog* sudah memenuhi *Definition of Done* maka *Increment* akan tercipta. *Definition of Done* menciptakan transparansi melalui pemberian pemahaman kepada setiap orang terkait pekerjaan yang sudah selesai dan menjadi bagian dari *Increment*. Untuk setiap butir *Product Backlog* yang dinyatakan tidak sesuai dengan *Definition of Done* tidak dapat dinyatakan selesai dan tidak dapat di-*release* sehingga akan dikembalikan kepada daftar *Product Backlog* untuk dilakukan pengerjaan kembali. *Developers* harus membiasakan diri dengan adanya *Definition of Done*, jika terdapat lebih dari tim Scrum dalam satu organisasi yang sama maka setiap tim Scrum akan memiliki sebuah *Definition of Done* yang sama.

Dalam penerapan Scrum tidak ada tuntutan untuk mengaplikasikan setiap tuntunan yang telah diberikan di atas, namun hasil dari pengaplikasian secara parsial tersebut tidak akan menghasilkan sebuah Scrum. Scrum bekerja sebagai sebuah wadah yang dapat menampung teknik, metode, atau praktek lain di dalamnya (Schwaber & Sutherland, 2020).

2.4 Pengembangan Aplikasi *Backend*

Melalui penjelasan yang diberikan oleh (Coursera, 2022), pengembangan aplikasi *backend* merupakan sebuah kegiatan atau pekerjaan yang dilakukan oleh seorang pengembang *backend*. Pengembang *backend* bertanggung jawab untuk membangun dan merawat mekanisme proses data serta menangani aksi-aksi dari aplikasi *frontend*.

Pengembang pada sisi *backend* melakukan pengembangan pada sisi *server* dari sebuah perangkat lunak (*software*), yang mana tidak dapat dilihat secara langsung dari depan. Pengembang *backend* memiliki peran untuk memastikan bahwa perangkat lunak berjalan dengan baik, focus terhadap pengelolaan basis data, logika *backend*, API, arsitektur aplikasi *backend*, dan pada *server*, apabila memungkinkan. Guna menangani semua hal tersebut, pengembang *backend* menggunakan bahasa pemrograman seperti C, Java, Go, JavaScript, dan yang lainnya. Umumnya, dalam sebuah tim pengembang perangkat lunak, pengembang *backend* akan berkolaborasi dengan pengembang *frontend*, manajer produk, *principal engineer*, dan *quality assurance* atau *software tester*.

Syarat utama untuk menjadi seorang pengembang *backend* adalah memiliki kemampuan teknis, berpikir analitik, dan kemampuan untuk berkolaborasi dengan baik. Hal-hal yang perlu dilakukan oleh pengembang *backend* sehari-hari antara lain adalah membangun dan merawat perangkat lunak, menulis kode pemrograman yang berkualitas, memecahkan permasalahan yang ada (*error*), melakukan *debugging*, dan memberikan dukungan pada keseluruhan tim pada proses pengembangan.

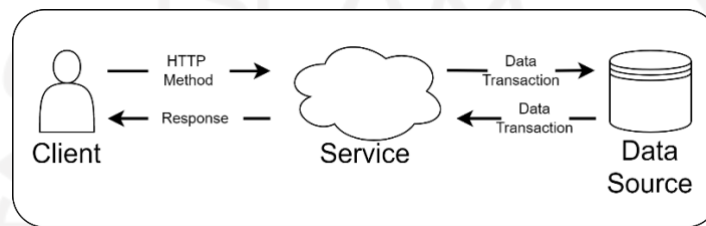
Teknologi yang digunakan oleh para pengembang *backend* tidak hanya terbatas pada pemakaian bahasa pemrograman dan basis data. Bahasa pemrograman yang biasanya digunakan untuk membuat sebuah aplikasi pada sisi *backend* umumnya adalah bahasa pemrograman Python, PHP, JavaScript, Ruby, Java, Go, dan Rust. Setiap bahasa pemrograman memiliki kelebihan dan kekurangan masing-masing, hal ini memungkinkan terdapat berbagai macam bentuk aplikasi *backend* yang cocok menggunakan bahasa pemrograman tertentu. Kemudian, pada sisi penyimpanan data menggunakan basis data yang sifatnya SQL dan NoSQL. Contohnya, pada pengembangan aplikasi *backend* Sajiloka digunakan basis data SQL, yang antara lain dapat digunakan adalah MySQL, PostgreSQL, CocroachDB, dan lain sebagainya. Sedangkan, untuk basis data NoSQL dapat menggunakan MongoDB, Firebase, DynamoDB, dan lain sejenisnya.

2.5 REST API

Application programming interface atau API merupakan antarmuka dari *libraries* berisi fungsi-fungsi yang dapat digunakan oleh *programmer*. (Stylos, Faulring, Yang, & Myers, 2009). Menurut laman resmi IBM yang membahas tentang API menyebutkan bahwa API merupakan sekumpulan aturan yang mendefinisikan bagaimana aplikasi dan perangkat dapat saling terhubung dan berkomunikasi. Secara singkat API memungkinkan sebuah aplikasi atau

service untuk dapat mengakses sumber daya dari *aplikasi* atau *service* lain. Aplikasi atau *service* yang melakukan akses disebut *client* sedangkan aplikasi atau *service* yang menyediakan sumber daya disebut *server*.

Pada implementasi nya terdapat beberapa arsitektur yang digunakan salah satunya adalah REST atau *representational state transfer*. REST API atau RESTful API merupakan istilah untuk API yang menerapkan arsitektur REST. Sehingga sebutan RESTful API memiliki makna yaitu API yang menerapkan REST secara keseluruhan.



Gambar 2.3 Alur ReST API

2.5.1 Prinsip Desain REST API

Berbeda dengan arsitektur lain seperti SOAP dan XML-RPC yang memberikan kerangka kerja cukup ketat bagi pengembang, REST API mungkin untuk dikembangkan dengan berbagai macam Bahasa pemrograman yang berbeda-beda dan mendukung banyak format data. Namun dengan fleksibilitas yang ditawarkan, terdapat juga *enam* prinsip dasar yang menjadi tetapan dalam arsitektur, yaitu *uniform interface*, *client-server decoupling*, *statelessness*, *cacheability*, *layered system architecture*, dan *code on demand* yang opsional.

a. *Uniform Interface*

Semua *request* API harus seragam walaupun berasal dari *client* yang berbeda. Sedangkan pada sisi *server* harus memastikan data yang dikirim konsisten, tidak kurang atau tidak berlebihan pada setiap *request* nya.

b. *Client-Server Decoupling*

Dalam REST API yang ideal, pihak *client* dan *server* harus independent atau tidak terikat oleh satu sama lain. Dalam implementasi nya sisi *client* hanya perlu mengetahui URI dari sumber daya yang diperlukan, sedangkan pada sisi *server* hanya perlu memberikan data sesuai yang diminta.

c. *Statelessness*

Arti dari prinsip ini adalah tidak menyimpan sesuatu atau tidak membentuk sebuah sesi. Sehingga pada penerapannya sebuah API yang menerapkan REST tidak boleh menyimpan data yang berhubungan dengan *request* dari *client*. Tugas dari *server* adalah hanya sebatas memberikan sumber daya yang dibutuhkan.

d. *Cacheability*

Jika memungkinkan sumber daya bisa disimpan dalam sebuah *cache* atau penyimpanan temporary. Penyimpanan dapat dilakukan pada sisi *client* maupun *server* dan pemberlakuan *caching* ini harus ada dalam sebuah persetujuan dari dua belah pihak. Tujuan dari dilakukannya *caching* adalah untuk meningkatkan performa pada sisi *client* dan juga meningkatkan skalabilitas pada sisi *server*.

e. *Layered System Architecture*

Dalam setiap komunikasi yang terjadi antara *client* dan *server* akan melalui beberapa lapisan sehingga antara *client* dan *server* tidak langsung berkomunikasi. Tujuan akhir dari dilakukannya sistem berlapis ini adalah membuat sisi *client* ataupun *server* tidak mengetahui sisi dalam dari mesin sehingga yang dilihat hanya lapisan luar.

f. *Code on Demand* (optional)

Pada umumnya sebuah REST API hanya mengirim sumber daya yang statis, namun dalam beberapa kasus sumber daya yang dikirim bisa berupa sebuah kode yang dapat dijalankan. Sehingga dalam kasus ini kode hanya dijalankan jika terdapat permintaan (*on demand*).

2.5.2 Cara Kerja REST API

REST API berkomunikasi menggunakan protokol HTTP yang terbagi menjadi dua jenis yaitu HTTP *Request* untuk *inbound* dan HTTP *Response* untuk *outbound*. Dalam HTTP *Request* REST API mendukung beberapa metode yang tersedia seperti POST, GET, PUT, DELETE. Fungsi umum dari HTTP *Request* adalah untuk melakukan manipulasi data dari *server* berupa Create, Read, Update, dan Delete atau lebih dikenal dengan CRUD.

Setelah melakukan *request* pihak *server* akan merespons dengan sebuah data atau sumber daya disebut sebagai *resource representation*. Respon data tersebut dikirim kepada *client* dengan menggunakan berbagai macam format. Format yang paling populer adalah JSON (JavaScript Object Notation) karena mudah dibaca oleh mesin maupun manusia, serta yang

paling utama adalah JSON tidak berkaitan dengan bahasa pemrograman apapun sehingga dapat digunakan oleh setiap bahasa pemrograman.

Terdapat sebuah bagian yang cukup penting dalam setiap *request* yang diberikan kepada *client* kepada *server* yaitu *request Header* dan *request* parameter. Kedua bagian *request* tersebut berperan memberikan informasi berupa metadata, otorisasi, URI, *caching*, *cookies* dan lain-lain.

Pada desain REST API yang ideal menurut IBM, setiap harus memiliki *Request Header* dan *Response Header* yang mengandung kode status HTTP (IBM Cloud Education, 2021).

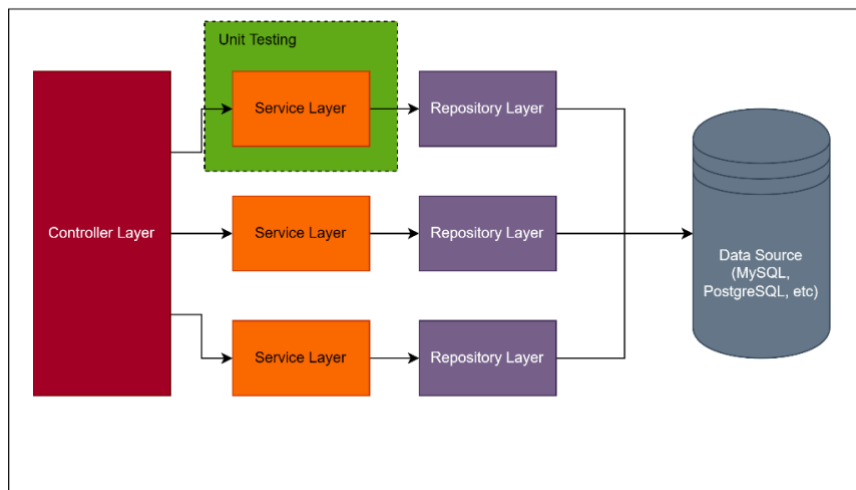
2.6 Pengujian Perangkat Lunak

Pada penelitian yang dilakukan oleh (Nidhra & Dondeti, 2012) pengujian perangkat lunak merupakan sebuah cara yang biasa dilakukan untuk memverifikasi dan memvalidasi suatu kualitas dari sebuah perangkat lunak. Kegiatan pengujian ini membutuhkan waktu yang cukup intensif dan biaya yang mahal. Pada pengujian perangkat lunak tradisional, teknik yang digunakan dalam pengujian dibagi menjadi dua yaitu pengujian *Blackbox* dan *Whitebox*. Pada istilah validasi dan verifikasi, pengujian *black box* digunakan untuk validasi sedangkan pengujian *white box* digunakan untuk verifikasi.

Pengujian perangkat lunak menggunakan *whitebox*. *Whitebox* disebut juga pengujian structural atau pengujian kotak kaca. Teknik pengujian structural membangun *test-case* berdasarkan data yang berasal dari *source code* yang dibangun. Sehingga penguji yang melakukan teknik pengujian ini memahami struktur *source code* dan untuk melakukan pengujian adalah dengan menuliskan kode *test-case* untuk mengeksekusi fungsi-fungsi dengan parameter tertentu. Fokus utama dari pengujian *Whitebox* adalah untuk menguji *control flow* dan data *flow* dari sebuah sistem.

Pengujian ini dibangun berbasis kode yang dilakukan oleh pengembang. Keseluruhan pengujian ini dilakukan untuk menguji setiap unit secara terpisah. Idealnya ukuran untuk unit itu sendiri sebatas fungsi atau method yang tidak lebih besar dari sebuah kelas. Semua proses *unit testing* mayoritas menggunakan *Mock Data* yang merupakan sebuah data palsu yang merepresentasikan ekspektasi hasil keluaran dari fungsi yang dilakukan pengujian (Mackinnon, Freeman, & Craig, 2001). Pada Gambar 2.4 ditunjukkan gambaran dari pengujian perangkat lunak yang dilakukan pada lapisan *service*.

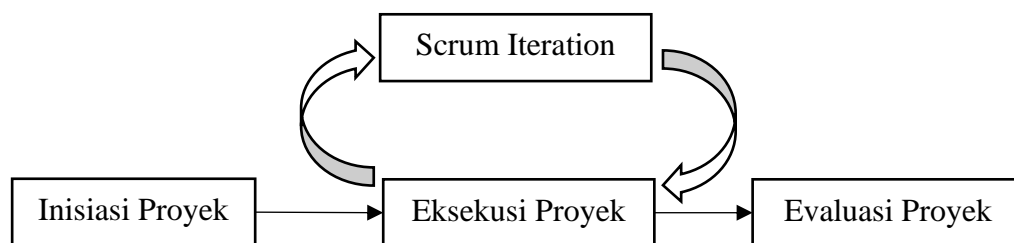
Application Programming Interface

Gambar 2.4 *Unit testing* pada GO

BAB III

METODE PENGEMBANGAN

Selama pengembangan yang dilakukan pada aplikasi Sajiloka, terdapat tiga tahapan utama, di antaranya inisiasi proyek, eksekusi proyek, dan evaluasi proyek. Alur metode pengembangan yang dilakukan ditunjukkan pada Gambar 3.1. Tahapan pertama dalam pengembangan adalah inisiasi proyek, yang mana membahas mengenai rencana dan rancangan awal dalam mempersiapkan seluruh kebutuhan pengembangan. Kemudian, tahapan kedua adalah eksekusi proyek, yang mana akan membahas mengenai kegiatan yang dilakukan selama pengembangan aplikasi *backend* yang menggunakan metode Scrum berlangsung. Tahapan terakhir adalah tahap evaluasi proyek, yang mana akan membahas mengenai pengujian kualitas produk yang telah dikembangkan pada aplikasi *backend* menggunakan *unit test*.



Gambar 3.1 Alur Metode Pengembangan

3.1 Inisiasi Proyek

Tahapan pertama dalam metode pengembangan adalah inisiasi proyek. Aplikasi Sajiloka merupakan sebuah hasil inisiasi selama mengikuti kegiatan perintisan bisnis pada 1000 SD. Bahasa pemrograman yang kemudian digunakan dalam pengembangan aplikasi *backend* Sajiloka adalah Bahasa Go (Golang). Selain bahasa pemrograman, digunakan arsitektur MVC (*Model View Controller*) pada aplikasi *backend* Sajiloka dengan memanfaatkan kerangka arsitektur kerja HTTP Gin Gonic. Selanjutnya, proses komunikasi yang berlangsung di antara *server frontend* dan *server backend* menggunakan format notasi JSON (*JavaScript Object Notation*), yang mana menerapkan standar komunikasi ReST (*Representational State Transfer*). Untuk memudahkan proses pengembangan aplikasi *backend* di-deploy pada sebuah layanan hosting gratis bernama Heroku. Aplikasi menggunakan layanan *versioning* GitHub yang langsung terhubung pada Heroku sehingga setiap melakukan *merge* ke *branch*

Development akan langsung dilakukan *deploy* otomatis atau istilah nya CI/CD (*Continuous Improvements / Continuous Developments*).

3.2 Eksekusi Proyek

Masuk ke dalam tahapan kedua dalam alur metode pengembangan Sajiloka, yaitu tahapan eksekusi proyek. Tahapan eksekusi proyek secara garis besar akan membahas mengenai kegiatan yang dilakukan selama pengembangan aplikasi *backend* yang menggunakan metode Scrum berlangsung. Pada tahapan ini, dilakukan tiga kali proses *Sprint*, yang kemudian diberi nama *Sprint 1*, *Sprint 2*, dan *Sprint 3*.

Pada setiap iterasi *Sprint* yang dilakukan, akan dijelaskan mengenai *task* yang perlu diselesaikan serta desain basis data terhadap *Sprint* tersebut. Pada *Sprint 1*, terdapat beberapa *task* yang di antaranya adalah halaman *login*, halaman pendaftaran, halaman depan (*home page*), dan halaman detail produk (*detail menu*). Kemudian, pada *Sprint 2*, terdapat dua *task* yang di antaranya adalah halaman bahan baku produk, dan halaman cara memasak produk. Terakhir, pada *Sprint 3*, terdapat empat *task* yang di antaranya adalah halaman daftar menu, halaman lihat detail menu, halaman tambah menu, dan halaman ubah menu. Keseluruhan detail akan dijelaskan secara detail pada sub-bab selanjutnya.

3.2.1 *Sprint 1*

Sprint 1 merupakan iterasi pertama dari kegiatan *Sprint* yang dilakukan selama tahap eksekusi proyek selama pengembangan aplikasi Sajiloka berlangsung. Ditunjukkan pada Gambar 3.2 daftar tugas-tugas yang akan dikerjakan selama waktu pembangunan aplikasi *backend* berlangsung. Di antara tugas-tugas yang akan dikerjakan selama *Sprint 1* berlangsung, yaitu halaman *login*, halaman pendaftaran, halaman depan (*home page*), dan halaman detail produk (*detail menu*).

Halaman *login* merupakan halaman yang bertugas sebagai “pintu” utama sebuah aplikasi. Sebelum pengguna dapat mengakses halaman detail produk, pengguna perlu menggunakan halaman *login* terlebih dahulu. Pada halaman *login*, Sajiloka mewajibkan penggunaannya untuk memasukan alamat surel yang sebelumnya telah didaftar kan pada saat pendaftaran dan *password*-nya.

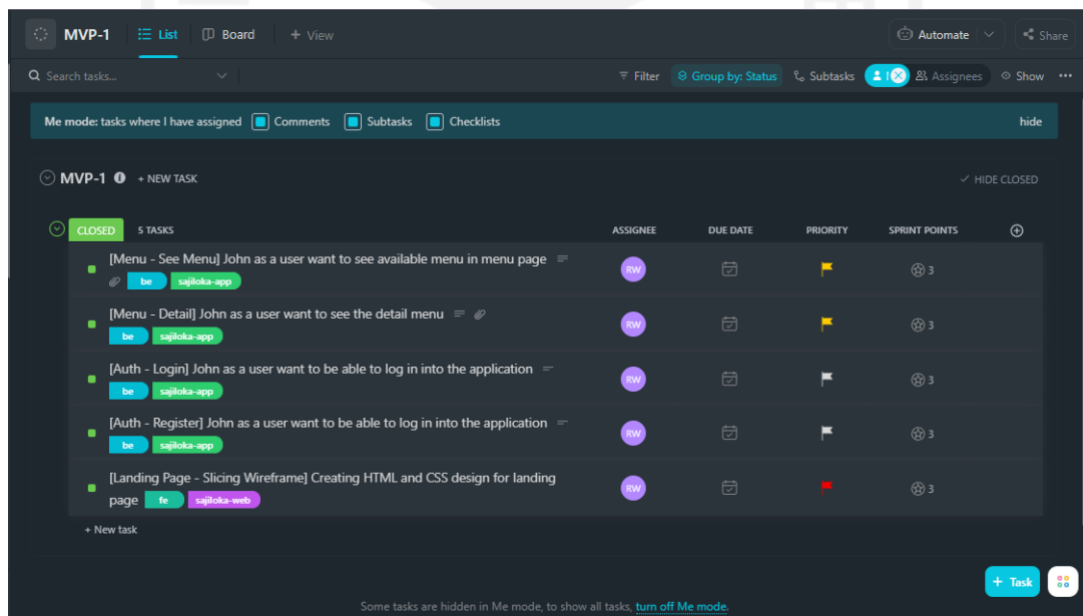
Halaman *login* merupakan halaman yang bertugas sebagai “pintu” utama sebuah aplikasi. Sebelum pengguna dapat mengakses halaman detail produk, pengguna perlu menggunakan halaman *login* terlebih dahulu. Pada halaman *login*, Sajiloka mewajibkan penggunanya untuk

memasukan alamat surel yang sebelumnya telah didaftar kan pada saat pendaftaran dan kata sandi nya.

Halaman kedua yang akan dikerjakan, adalah halaman pendaftaran atau dapat disebut juga sebagai halaman registrasi. Sajiloka mewajibkan pengguna untuk mendaftarkan diri untuk dapat melihat konten-konten yang disediakan. Beberapa hal yang diperlukan ketika mendaftarkan diri adalah nama pengguna, alamat surel pengguna, nomor telepon pengguna, dan *password* pengguna.

Kemudian, pada halaman depan Sajiloka, disajikan informasi mengenai informasi Sajiloka dan keseluruhan produk yang ada melalui kategorisasi. Tugas terakhir yang dikerjakan adalah halaman detail produk. Halaman detail produk pada Sajiloka akan menampilkan informasi detail mengenai menu-menu yang tersedia.

Pada Gambar 3.2 diperlihatkan informasi mengenai tugas yang akan dikerjakan pada *Sprint 1*. Untuk manajemen tugas-tugas digunakan aplikasi ClickUp. Pada setiap tugas yang diletakkan di ClickUp, tugas akan dapat ditujukan kepada *hacker*, tugas dapat diberi *deadline* pengerjaan, prioritas pada tugas juga dapat ditentukan berdasarkan warna bendera yang ada, dan yang terakhir adalah jumlah dari *sprint points* pada setiap tugas yang ada.



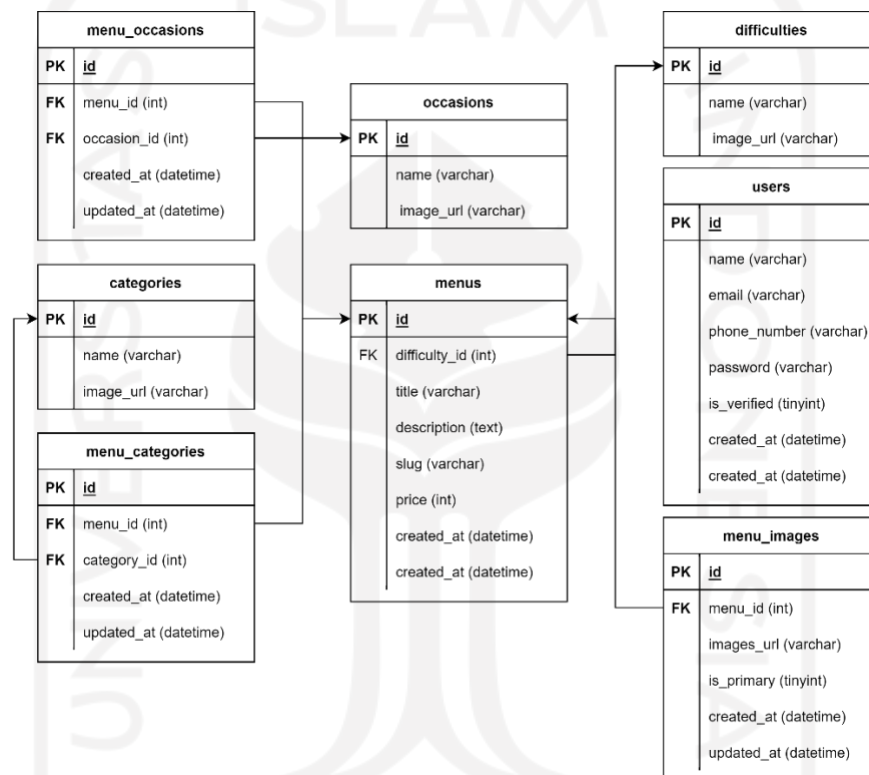
Task Description	Assignee	Due Date	Priority	Sprint Points
[Menu - See Menu] John as a user want to see available menu in menu page	John	2023-10-10	High	3
[Menu - Detail] John as a user want to see the detail menu	John	2023-10-10	High	3
[Auth - Login] John as a user want to be able to log in into the application	John	2023-10-10	Medium	3
[Auth - Register] John as a user want to be able to log in into the application	John	2023-10-10	Medium	3
[Landing Page - Slicing Wireframe] Creating HTML and CSS design for landing page	John	2023-10-10	Low	3

Gambar 3.2 Task Sprint 1

Melalui *user story* yang ada pada task yang tertera pada Gambar 3.2, maka dibuat sebuah desain tabel basis data yang ditunjukkan secara detail pada Gambar 3.3. Terdapat 8 *table* yang

dibuat pada Gambar 3.3 guna mendukung kebutuhan basis data terhadap fitur-fitur pada *Sprint 1*, di antaranya *table users*, *menus*, dan *menu_images*.

Table users merupakan sebuah *table* yang bertujuan untuk menyimpan data pengguna, seperti nama pengguna (*name*), alamat surel pengguna (*email*), nomor telepon pengguna (*phone_number*), dan kata sandi (*password*). *Table users* akan digunakan pada proses *login*, *register*, dan akan digunakan juga untuk kebutuhan otorisasi Ketika pengguna melakukan proses transaksi pada aplikasi Sajiloka.



Gambar 3.3 Desain basis data *Sprint 1*

Table menus merupakan sebuah *table* bertujuan untuk menyimpan data dari produk, seperti tingkat kesulitan yang direferensikan menggunakan *foreign key difficulty_id*, nama produk (*title*), deskripsi produk (*description*), *identifier slug* yang mana identifikasi yang disisipkan pada URL, dan harga produk (*price*). Sebagai contoh sebuah *identifier slug* adalah *sajiloka.com/menus/nasi-padang-bal-1*. *Table menu_images* merupakan sebuah *table* yang memiliki data milik *menus* dengan kardinalitas *one-to-many* yang menyimpan gambar-gambar produk. Kardinalitas *one-to-many* pada *table* tersebut diartikan bahwa setiap satu menu memiliki banyak gambar produk.

Table occasions dan *categories* akan menyimpan daftar kejadian dan kategori dari produk-produk yang nantinya akan dihubungkan dengan *menus*. Kardinalitas yang digunakan pada kedua *table* tersebut adalah *many-to-many*, yang diartikan bahwa banyak *categories* atau *occasions* memiliki atau dimiliki oleh banyak menu. Kardinalitas tersebut terjadi melalui *table menu_categories* dan *menu_occasions*. *Table difficulties* akan menyimpan tingkat kesulitan dari *menus*, yang direferensikan melalui *foreign key difficulty_id* dengan kardinalitas *one-to-many*. Kardinalitas *one-to-many* pada *table* tersebut diartikan bahwa setiap satu *difficulty* memiliki banyak menu.

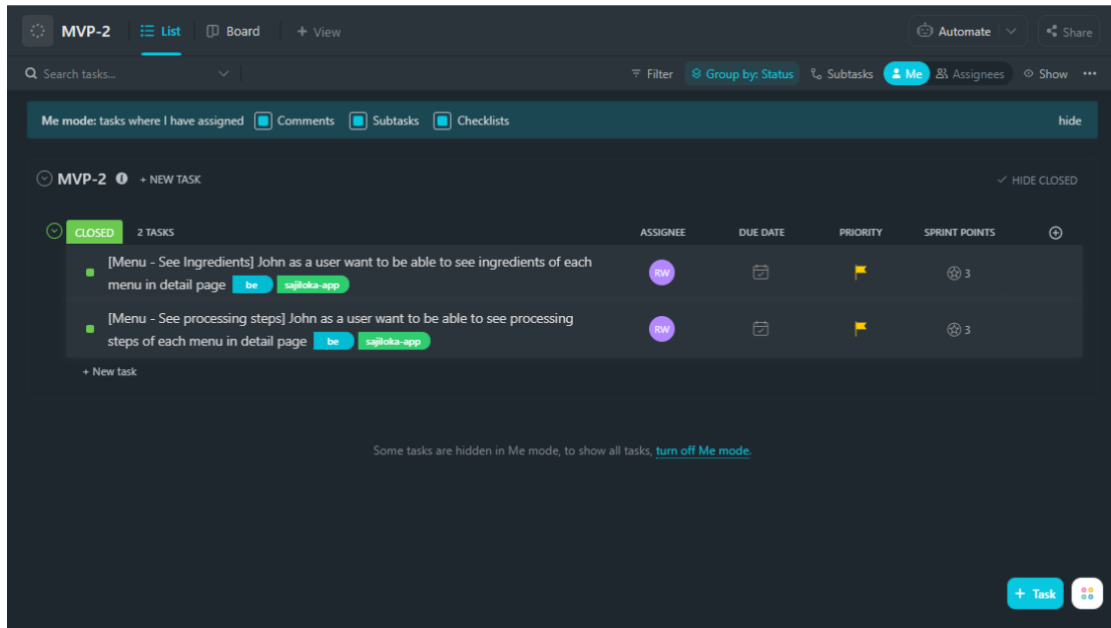
3.2.2 *Sprint 2*

Sprint 2 merupakan iterasi kedua dari kegiatan *Sprint* yang dilakukan selama tahap eksekusi proyek selama pengembangan aplikasi Sajiloka berlangsung. Ditunjukkan pada Gambar 3.4 daftar tugas-tugas yang akan dikerjakan selama waktu pembangunan aplikasi *backend* berlangsung. Di antara tugas-tugas yang akan dikerjakakan selama *Sprint 2* berlangsung, yaitu halaman lihat bahan baku produk, dan halaman lihat cara memasak produk.

Halaman pertama yang akan dibuat pada *Sprint 2* adalah halaman lihat bahan baku produk. Halaman lihat bahan baku produk merupakan bagian dari halaman detail produk yang menampilkan daftar bahan baku yang digunakan untuk mengolah produk.

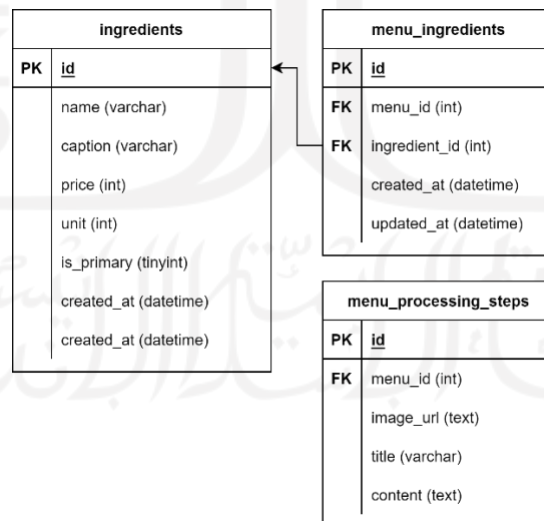
Kemudian, halaman kedua yang dibuat pada *Sprint 2* adalah bagian lihat cara memasak produk. Halaman lihat cara memasak produk merupakan bagian dari detail produk yang menampilkan daftar langkah-langkah pengolahan dari produk. Langkah-langkah yang ditampilkan pada setiap produk dimulai dari langkah persiapan memasak hingga langkah menghadirkan.

Pada Gambar 3.4 diperlihatkan informasi mengenai tugas yang akan dikerjakan pada *Sprint 2*. Pada setiap tugas yang diletakkan di ClickUp, tugas dapat ditujukan kepada *hacker*, tugas dapat diberi *deadline* pengerjaan, prioritas pada tugas juga dapat ditentukan berdasarkan warna bendera, dan yang terakhir adalah jumlah dari *sprint points* pada setiap tugas yang ada.



Gambar 3.4 Task Sprint 2

Melalui *user story* yang ada pada task yang tertera pada Gambar 3.5, maka dibuat sebuah desain tabel basis data yang ditunjukkan secara detail pada Gambar 3.5. Terdapat 3 *table* yang dibuat pada Gambar 3.5 guna mendukung kebutuhan basis data terhadap fitur-fitur pada *Sprint 1*, di antaranya *ingredients*, *menu_ingredients*, dan *menu_processing_steps*.



Gambar 3.5 Desain basis data Sprint 2

Table ingredients merupakan sebuah *table* yang bertujuan untuk menyimpan seluruh data bahan baku, seperti nama (*name*), deskripsi (*caption*), harga (*price*), dan takaran (*unit*).

Table menu_ingredients merupakan sebuah *table* yang bertujuan untuk menghubungkan produk (*menus*) dengan bahan bakunya (*ingredients*). Kardinalitas *many-to-many* pada hubungan kedua *table* tersebut diartikan bahwa banyak produk dapat dimiliki atau memiliki banyak menu. *Table menu_processing_steps* merupakan sebuah *table* yang bertujuan untuk menyimpan seluruh data cara pengolahan dari produk menggunakan kardinalitas *one-to-many*. Data-data yang disimpan di antaranya adalah gambar (*image_url*), judul (*title*), dan konten (*content*). Kardinalitas *one-to-many* diartikan bahwa setiap satu menu memiliki banyak langkah memasak produk.

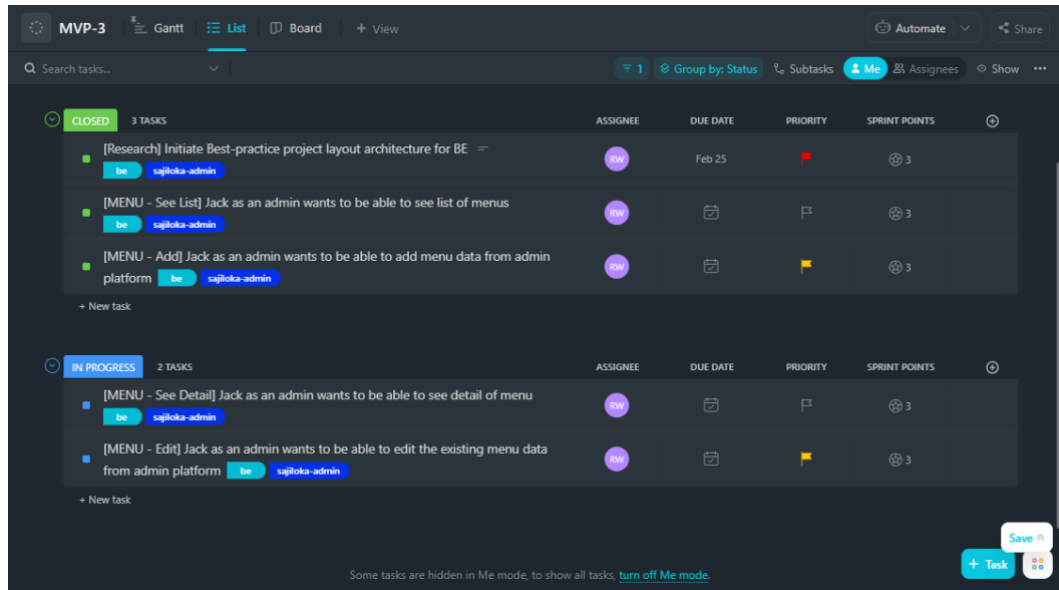
3.2.3 *Sprint 3*

Sprint 3 merupakan iterasi ketiga dari kegiatan *Sprint* yang dilakukan selama tahap eksekusi proyek selama pengembangan aplikasi Sajiloka berlangsung. Ditunjukkan pada Gambar 3.6 daftar tugas-tugas yang akan dikerjakan selama waktu pembangunan aplikasi *backend* berlangsung. Di antara tugas-tugas yang akan dikerjakan selama *Sprint 3* berlangsung, yaitu halaman daftar menu, halaman lihat detail menu, halaman tambah menu, dan halaman ubah menu.

Halaman daftar menu merupakan halaman pertama yang perlu dikerjakan pada *Sprint 3*. Halaman daftar menu akan menunjukkan daftar dari produk-produk yang telah ada dalam basis data. Kemudian, halaman kedua yang akan dikerjakan, adalah halaman detail menu. Halaman detail menu akan menunjukkan detail dari produk-produk yang ada.

Selanjutnya, halaman yang akan dikerjakan pada *Sprint 3* adalah halaman tambah menu. Halaman ini bertujuan untuk menambahkan produk ke dalam basis data. Halaman terakhir yang akan dibuat adalah halaman ubah menu. Halaman ubah menu merupakan halaman yang digunakan untuk melakukan perubahan data produk yang telah ada di basis data.

Pada Gambar 3.6 diperlihatkan informasi mengenai tugas yang akan dikerjakan pada *Sprint 3*. Pada setiap tugas yang diletakkan di ClickUp, tugas dapat ditujukan kepada *hacker*, tugas dapat diberi *deadline* pengerjaan, prioritas pada tugas juga dapat ditentukan berdasarkan warna bendera, dan yang terakhir adalah jumlah dari *sprint points* pada setiap tugas yang ada.



Gambar 3.6 Task Sprint 3

Aplikasi *backend* pada sisi admin menggunakan basis data yang sama seperti basis data yang digunakan pada aplikasi *backend* pada sisi admin, sehingga tidak terdapat basis data penting yang baru.

Sayang nya, tidak seluruh tugas dapat selesai pengerjaan nya. Terdapat beberapa tugas yang tidak dapat diselesaikan, di antaranya adalah tugas pada halaman tambah menu dan halaman ubah menu. Tugas halaman tambah menu hanya mencapai 50% pengerjaan. Sedangkan, tugas halaman ubah menu tidak sempat dikerjakan karena kegiatan *Sprint 3* dan masa inkubasi telah selesai, yang mana tim Sajiloka dinyatakan tidak lolos.

3.3 Evaluasi Proyek

Tahapan evaluasi proyek dilakukan setelah seluruh kegiatan eksekusi proyek selesai. Evaluasi proyek bertujuan untuk melakukan validasi terhadap kualitas dari aplikasi yang telah dibangun. Evaluasi proyek dilakukan untuk menguji kualitas aplikasi *backend* menggunakan pengujian *whitebox unit test* yang telah disediakan oleh bahasa pemrograman Go.

Pengujian yang dilakukan pada evaluasi proyek meliputi seluruh fungsi-fungsi yang ada pada setiap *service* dari aplikasi *backend*. Pengujian dibagi dua yaitu unit yang berada pada sisi pengguna atau *Consumer* dan unit yang berada pada sisi pengelola atau *Admin*.

3.3.1 Auth Service

Auth service merupakan *service* yang digunakan untuk proses autentikasi. Pada *service* ini, terdapat dua fungsi yang akan dilakukan *unit testing*, antara lain adalah fungsi *GenerateToken* dan *ValidateToken*.

Pertama, fungsi *GenerateToken* bertujuan untuk melakukan *generate* token JWT (JSON Web Token), yang mana akan didapatkan oleh pengguna saat melakukan *login* pada aplikasi Sajiloka. *Unit testing* yang dilakukan pada fungsi ini ditunjukkan pada Gambar 3.7, yang mana bertujuan untuk melakukan pengecekan terhadap *value* pada setiap variabel token yang telah di-*generate* oleh fungsi agar tidak kosong.

```
func TestGenerateToken(t *testing.T) {
    testService := NewService()

    testToken := testService.GenerateToken(1)
    testTokenLst := strings.Split(testToken, ".")

    assert.NotNil(t, testToken)
    assert.Equal(t, len(testTokenLst), 3)
    assert.Equal(t, testTokenLst[0], "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9")
    assert.Equal(t, testTokenLst[1], "eyJ1c2VyX2lkIjoxfQ")
}
```

Gambar 3.7 Unit testing fungsi *GenerateToken* (*Auth Service*)

Kedua, fungsi *ValidateToken* bertujuan untuk melakukan validasi token JWT yang telah diterima berdasarkan kesesuaian *secret key* yang telah ditetapkan. *Unit testing* pada fungsi ini ditunjukkan pada Gambar 3.8, yang bertujuan untuk melakukan pengecekan terhadap hasil dari validasi token. Pengecekan yang dilakukan di antaranya ialah guna memastikan bahwa token tidak kosong, proses *claim* token berhasil, token bernilai valid, isi dari *claim* token tidak kosong, dan isi dari *claim* token yang berupa *user_id* sesuai dengan yang terdapat pada token.

```
func TestValidateToken(t *testing.T) {
    testService := NewService()

    token := testService.ValidateToken("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxfQ.N1c2nJo8nCgnpvmfldYSNq_0tqzl9gs_SBG_aDhVCkE")
    claim, ok := token.Claims.(jwt.MapClaims)
    userId := claim["user_id"].(float64)

    assert.NotNil(t, token)
    assert.True(t, ok, "validation error")
    assert.True(t, token.Valid, "validation error")
    assert.NotNil(t, claim)
    assert.Equal(t, userId, float64(1))
}
```

Gambar 3.8 Unit testing fungsi *ValidateToken* (Auth Service)

3.3.2 Menu Service

Menu service adalah sebuah *service* yang digunakan untuk melakukan pemanggilan ke *database* untuk mendapatkan data *menu*. Terdapat tiga fungsi yang akan dilakukan *unit testing* pada *service* ini, di antaranya adalah fungsi *GetAllMenu*, *GetMenuById*, dan *GetMenuProcessingStepsById*.

Pertama, fungsi *GetAllMenu* bertujuan untuk mengambil seluruh data *menu* yang berada di dalam *database*. *Unit testing* pada fungsi ini ditunjukkan pada Gambar 3.9, yang bertujuan untuk melakukan pengecekan terhadap koneksi *database* dan *mock data*, yang mana direpresentasikan dengan variable *menuA*, yang sebelumnya telah didefinisikan sesuai dengan *output* yang diberikan fungsi *GetAllMenu*.

```
func TestGetAllMenu(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    menuA := Menu{
        ID: 1,
        DifficultyID: 1,
        Title: "Title1",
        Description: "desc2",
        Slug: "sluch-1",
        Price: 1,
        Duration: 1,
        Portion: 1,
        CreatedAt: sql.NullTime{},
        UpdatedAt: sql.NullTime{},
        MenuImages: []MenuImage{},
        MenuOccasions: []occasion.Occasion{},
        MenuCategories: []category.Category{},
    }

    mockRepo.On("FindAll").Return([]Menu{menuA}, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetAllMenu()
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data[0].ID)
}
```

Gambar 3.9 Unit testing fungsi *GetAllMenu* (Menu Service)

Kemudian, fungsi *GetMenuById* bertujuan untuk mengambil data *menu* dari *database* berdasarkan *id* yang telah diberikan. *Unit testing* yang dilakukan pada fungsi ini ditunjukkan

pada Gambar 3.10, yang mana bertujuan untuk melakukan pengecekan kesamaan terhadap *mock data* yang telah direpresentasikan dengan variable *menuA* dengan *output* dari fungsi *GetMenuById*.

```
func TestGetMenuById(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    menuA := Menu{
        ID:          1,
        DifficultyID: 1,
        Title:        "Title1",
        Description:  "desc2",
        Slug:         "sluch-1",
        Price:        1,
        Duration:     1,
        Portion:       1,
        CreatedAt:    sql.NullTime{},
        UpdatedAt:    sql.NullTime{},
        MenuImages:   []MenuImage{},
        MenuOccasions: []occasion.Occasion{},
        MenuCategories: []category.Category{},
    }

    mockRepo.On("FindById").Return(menuA, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetMenuById(GetMenuDetailInput{ID: 1})
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data.ID)
}
```

Gambar 3.10 Unit testing fungsi *GetMenuById* (Menu Service)

Terakhir, fungsi *GetMenuProcessingStepsById* bertujuan untuk mengambil seluruh cara memasak yang berada pada *menu* melalui *id*. Unit testing yang dilakukan pada fungsi ini ditunjukkan pada Gambar 3.11, yang mana bertujuan untuk melakukan pengecekan kesesuaian *mock data* yang telah direpresentasikan dengan variable *processingStepA* dengan *output* dari fungsi *GetMenuProcessingStepsById*.

```
func TestGetProcessingStepsById(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    processingStepA := MenuProcessingStep{
        ID:          1,
        MenuID: 1,
```



```

    }

    mockRepo.On("FindProcessingStepsByMenuId").Return([]MenuProcessingStep{processingStepA}, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetProcessingStepsById(GetMenuDetailInput{ID: 1})
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data[0].ID)
    assert.Equal(t, 1, data[0].MenuID)
}

```

Gambar 3.11 Unit testing fungsi *GetProcessingStepsById* (Menu Service)

3.3.3 Ingredient Service

Ingredient service merupakan sebuah *service* yang digunakan untuk mengetahui semua data bahan masak dari *database*. Hanya terdapat satu fungsi yang akan dilakukan *unit testing*, yaitu *GetMenuIngredients*.

Fungsi *GetMenuIngredients* bertujuan untuk mendapatkan semua data *ingredients* dari *database* pada *menu* melalui *id*. *Unit testing* yang dilakukan pada fungsi ini ditunjukkan pada Gambar 3.12, yang mana bertujuan untuk melakukan pengecekan kesesuaian terhadap *mock data* yang direpresentasikan dengan variable *ingredientA* terhadap *output* dari fungsi *GetMenuIngredients*.

```

func TestGetMenuIngredients(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    ingredientA := Ingredient{
        ID: 1,
        Name: "bahan-1",
    }

    mockRepo.On("FindByMenuId").Return([]Ingredient{ingredientA}, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetMenuIngredients(1)
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data[0].ID)
    assert.Equal(t, "bahan-1", data[0].Name)
}

```

Gambar 3.12 Unit testing fungsi *GetMenuIngredients* (Ingredient Service)

3.3.4 Category Service

Category service merupakan *service* yang digunakan untuk mengetahui semua data *category* dari *database*. Pada *service* ini terdapat dua fungsi yang akan dilakukan *unit testing*, yaitu *GetAllCategories* dan *GetAllMenuById*.

Pertama, fungsi *GetAllCategories* bertujuan untuk mendapatkan semua data kategori dari *database*. *Unit testing* yang akan dilakukan pada fungsi ini ditunjukkan pada Gambar 3.13, yang bertujuan untuk melakukan pengecekan kesamaan terhadap *mock data* yang direpresentasikan dengan variable *categoryA* terhadap *output* dari fungsi *GetAllCategories*.

```
func TestGetAllCategories(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    categoryA := Category{
        ID: 1,
        Name: "kategori-1",
    }

    mockRepo.On("FindAll").Return([]Category{categoryA}, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetAllCategories()
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data[0].ID)
    assert.Equal(t, "kategori-1", data[0].Name)
}
```

Gambar 3.13 Unit testing fungsi *GetAllCategories* (Category Service)

Kemudian, fungsi *GetAllMenuById* bertujuan untuk mendapatkan semua data *Category* dari *database* pada *menu* melalui id. *Unit testing* yang dilakukan pada fungsi ini ditunjukkan pada Gambar 3.14, yang bertujuan untuk melakukan pengecekan kesesuaian terhadap *mock data* yang direpresentasikan dengan variable *categoryA* terhadap *output* dari fungsi *GetAllMenuById*.

```
func TestGetAllByMenuID(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()
```

```

categoryA := Category{
    ID: 1,
    Name: "kategori-1",
}

mockRepo.On("FindByMenuID").Return([]Category{categoryA}, nil)
testService := NewService(mockRepo, db, validator)
data := testService.GetAllByMenuID(1)
mockRepo.AssertExpectations(t)

assert.NotNil(t, data, "Data is nil")
assert.Equal(t, 1, data[0].ID)
assert.Equal(t, "kategori-1", data[0].Name)
}

```

Gambar 3.14 Unit testing fungsi *GetAllByMenuId* (Category Service)

3.3.5 Occasion Service

Occasion service merupakan sebuah *service* yang digunakan untuk mengetahui semua data *occasion* dari *database*. Pada *service* ini terdapat dua fungsi yang akan dilakukan unit testing, yaitu *GetAllOccasions* dan *GetAllMenuById*.

Fungsi *GetAllOccasions* bertujuan untuk mendapatkan semua data *occasion* dari *database*. Unit testing yang akan dilakukan pada fungsi ini ditunjukkan pada Gambar 3.15, yang bertujuan untuk melakukan pengecekan kesamaan terhadap *mock data* yang direpresentasikan dengan variable *occasionA* terhadap *output* dari fungsi *GetAllOccasions*.

```

func TestGetAllOccasions(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    occasionA := Occasion{
        ID: 1,
        Name: "occasion-1",
    }

    mockRepo.On("FindAll").Return([]Occasion{occasionA}, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetAllOccasions()
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data[0].ID)
    assert.Equal(t, "occasion-1", data[0].Name)
}

```

Gambar 3.15 Unit testing fungsi *GetAllOccasions* (Occasion Service)

Fungsi *GetAllByMenuId* bertujuan untuk mendapatkan semua data *occasion* dari *database* pada *menu* melalui *id*. Unit testing yang dilakukan pada fungsi ini ditunjukkan pada

Gambar 3.16, yang bertujuan untuk melakukan pengecekan kesesuaian terhadap *mock data* yang direpresentasikan dengan variable *occasionA* terhadap *output* dari fungsi *GetAllByMenuId*.

```
func TestGetAllByMenuID(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    occasionA := Occasion{
        ID: 1,
        Name: "occasion-1",
    }

    mockRepo.On("FindByMenuID").Return([]Occasion{occasionA}, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetAllByMenuID(1)
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data[0].ID)
    assert.Equal(t, "occasion-1", data[0].Name)
}
```

Gambar 3.16 Unit testing fungsi *GetAllByMenuId* (*Occasion Service*)

3.3.6 Difficulty Service

Difficulty service merupakan sebuah *service* yang digunakan untuk mengetahui semua data *difficulty* dari *database*. Pada *service* ini terdapat dua fungsi yang akan dilakukan *unit testing*, yaitu *GetAllDifficulty* dan *GetById*.

Fungsi *GetAllDifficulty* bertujuan untuk mendapatkan semua data *difficulty* dari *database*. *Unit testing* yang akan dilakukan pada fungsi ini ditunjukkan pada Gambar 3.17, yang bertujuan untuk melakukan pengecekan kesamaan terhadap *mock data* yang direpresentasikan dengan variable *difficultyA* terhadap *output* dari fungsi *GetAllDifficulty*.

```
func TestGetAllDifficulties(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    difficultyA := Difficulty{
        ID: 1,
        Name: "difficulty-1",
    }

    mockRepo.On("FindAll").Return([]Difficulty{difficultyA}, nil)
```

```

testService := NewService(mockRepo, db, validator)
data := testService.GetAllDifficulties()
mockRepo.AssertExpectations(t)

assert.NotNil(t, data, "Data is nil")
assert.Equal(t, 1, data[0].ID)
assert.Equal(t, "difficulty-1", data[0].Name)
}

```

Gambar 3.17 Unit testing fungsi *GetAllDifficulties* (Difficulty Service)

Fungsi *GetById* bertujuan untuk mendapatkan semua data *difficulty* dari *database* berdasarkan *id*. Unit testing yang dilakukan pada fungsi ini ditunjukkan pada Gambar 3.18, yang bertujuan untuk melakukan pengecekan kesesuaian terhadap *mock data* yang direpresentasikan dengan variable *difficultyA* terhadap *output* dari fungsi *GetById*.

```

func TestGetById(t *testing.T) {
    mockRepo := new(MockRepository)
    db, err := sql.Open(dbDriver, dbCfg)
    if err != nil {
        log.Fatal("cannot connect to db:", err)
    }

    validator := validator.New()

    difficultyA := Difficulty{
        ID: 1,
        Name: "difficulty-1",
    }

    mockRepo.On("FindById").Return(difficultyA, nil)
    testService := NewService(mockRepo, db, validator)
    data := testService.GetById(1)
    mockRepo.AssertExpectations(t)

    assert.NotNil(t, data, "Data is nil")
    assert.Equal(t, 1, data.ID)
    assert.Equal(t, "difficulty-1", data.Name)
}

```

Gambar 3.18 Unit testing fungsi *GetById* (Difficulty Service)

3.3.7 Menu Service (Admin)

Menu service pengelola merupakan sebuah *service* yang digunakan untuk mengelola semua data *Menu* pada *database*. *Service* ini masuk ke dalam pengujian aplikasi *backend* sisi pengelola atau *Admin*. Pada *service* ini hanya terdapat satu fungsi yang akan dilakukan *unit testing*, yaitu *CreateMenu*.

Fungsi *CreateMenu* bertujuan untuk membuat data *menu* pada *database*. Unit testing yang akan dilakukan pada fungsi ini ditunjukkan pada Gambar 3.19, yang bertujuan untuk melakukan pengecekan kesamaan terhadap *mock data* dengan *output* dari fungsi *CreateMenu*.

```

func TestCreateMenu(t *testing.T) {
    mockRepo := new(MockRepository)

```

```

db, err := sql.Open(dbDriver, dbCfg)
if err != nil {
    log.Fatal("cannot connect to db:", err)
}

menuA := Menu{
    ID:          1,
    DifficultyID: 1,
    Title:       "Title1",
    Description: "desc2",
    Slug:        "sluch-1",
    Price:       1,
    Duration:    1,
    Portion:     1,
    CreatedAt:   sql.NullTime{},
    UpdatedAt:   sql.NullTime{},
    MenuImages: []MenuImage{},
}

input := CreateMenuInput{
    Title:       "Title1",
    Description: "desc2",
    Duration:    1,
    Portion:     1,
    Price:       1,
    DifficultyID: 1,
}

validator := validator.New()
mockRepo.On("Save").Return(menuA, nil)
testService := NewService(mockRepo, db, validator)
data := testService.CreateMenu(input)
mockRepo.AssertExpectations(t)

assert.NotNil(t, data, "Data is nil")
assert.Equal(t, input.Title, data.Title)
}

```

Gambar 3.19 *Unit testing* fungsi CreateMenu (Menu Service)

BAB IV

HASIL DAN REFLEKSI

Bab ini akan membahas secara detail mengenai hasil dan refleksi yang didapatkan. Adapun hasil yang akan dibahas, terbatas pada hasil pengembangan aplikasi dan hasil pengujian aplikasi. Sedangkan, refleksi yang akan dibahas pada bab ini terbatas pada kendala dan hambatan yang dialami, tantangan yang dihadapi, capaian yang didapatkan, dan wawasan kegiatan yang didapatkan selama pengembangan aplikasi berlangsung.

4.1 Hasil Pengembangan Aplikasi

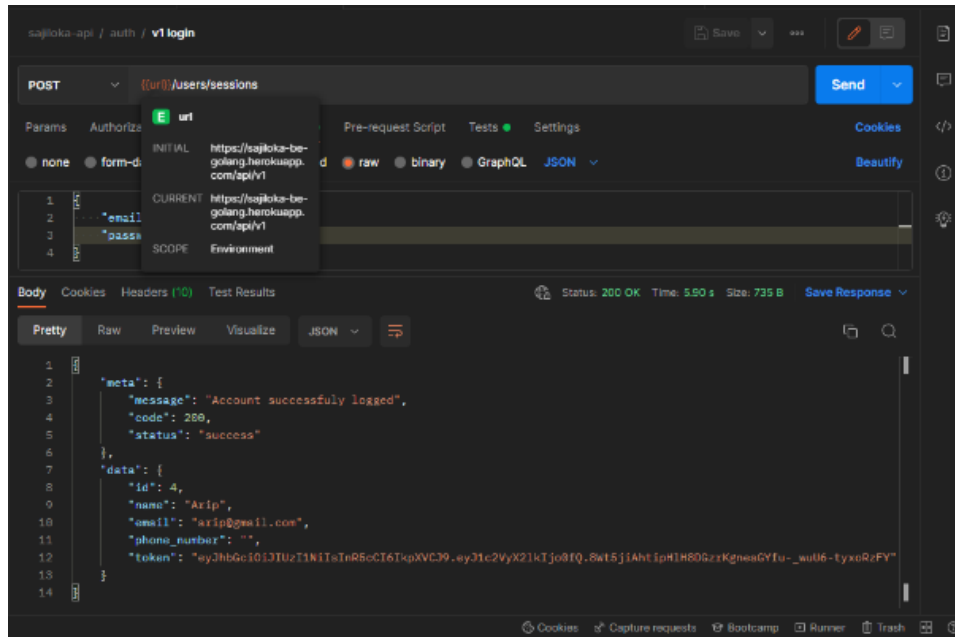
Pengembangan aplikasi yang dilakukan pada sisi *backend* dilakukan menggunakan bahasa GO. Hasil dari pengembangan aplikasi yang dikembangkan berupa sekumpulan API yang mendukung kebutuhan data dan transaksi data aplikasi pada sisi *frontend*. Terdapat enam halaman yang dihasilkan dari pengembangan aplikasi menggunakan Postman dan beberapa di antaranya hanya dapat diakses oleh admin saja. Halaman yang telah dibuat antara lain adalah halaman autentikasi, halaman utama, halaman detail produk, halaman daftar produk dari sisi admin, halaman detail produk dari sisi admin, dan halaman tambah produk dari sisi admin.

Berikut akan disajikan hasil API yang dipanggil melalui Postman dan telah dikategorikan berdasarkan halamannya.

a. API Autentikasi

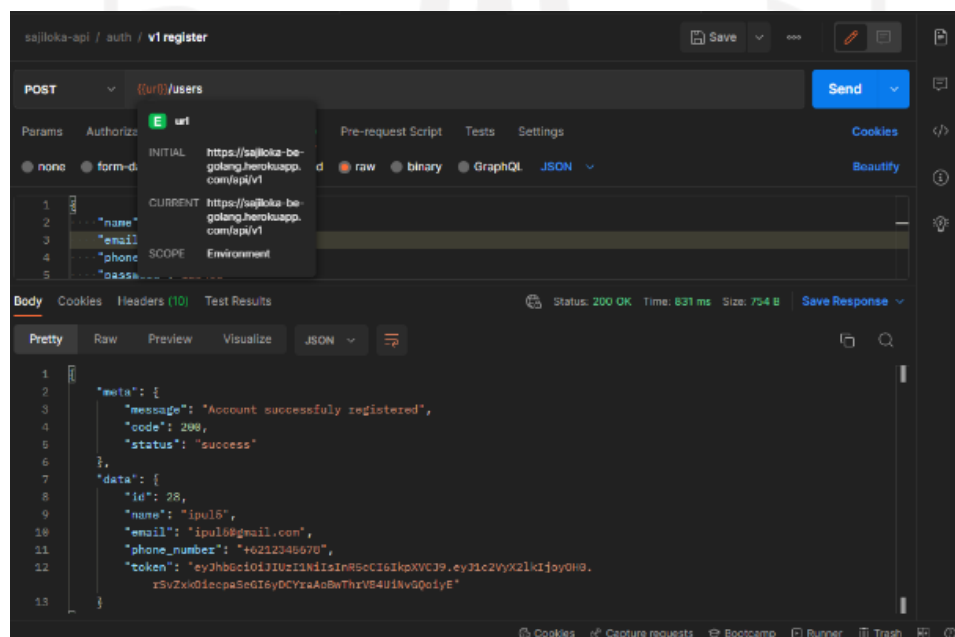
Pada bagian ini, API Autentikasi mendukung proses autentikasi pengguna pada halaman masuk (*login*), halaman daftar (*register*), dan halaman *get own account* atau mendapatkan akun pengguna melalui token. Ketiga API Autentikasi ditunjukkan pada Gambar 4.1, Gambar 4.2, dan Gambar 4.3.

Halaman masuk (*login*) ditunjukkan pada Gambar 4.1. Pada hasil yang ditunjukkan menggunakan JSON, terdapat dua hasil yakni *meta* dan *data*. Pada hasil *meta*, terdapat *key message* yang memberikan pesan bahwa akun telah berhasil masuk, dan *key status* memberikan status sukses. Sedangkan, pada hasil *data*, memberikan informasi detail terkait pengguna yang telah sukses masuk. Informasi detail yang diberikan di antaranya adalah ID pengguna, nama, alamat surel, nomor telepon, dan token.



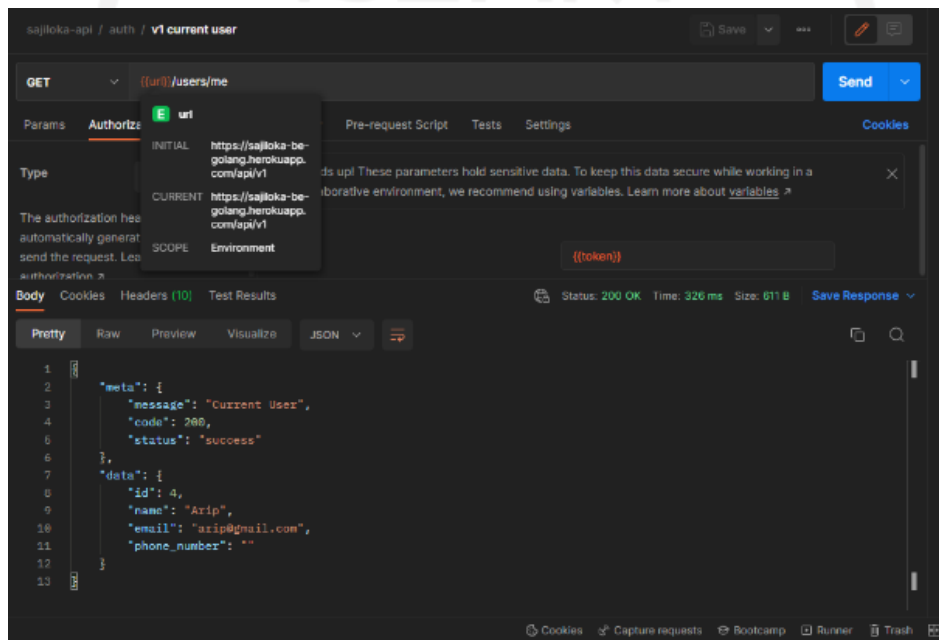
Gambar 4.1 API Masuk Pengguna

Halaman daftar (*register*) ditunjukkan pada Gambar 4.2. Pada hasil yang ditunjukkan menggunakan JSON, terdapat dua hasil yakni *meta* dan *data*. Pada hasil *meta*, terdapat *key message* yang memberikan pesan bahwa akun telah berhasil melakukan pendaftaran, dan *key status* memberikan status sukses. Sedangkan, pada hasil *data*, memberikan informasi detail terkait pengguna yang telah sukses masuk. Informasi detail yang diberikan di antaranya adalah ID pengguna, nama, alamat surel, nomor telepon, dan token.



Gambar 4.2 API Daftar Pengguna

Halaman dapat kan akun pengguna (*get own account*) ditunjukkan pada Gambar 4.3. Pada hasil yang ditunjukkan menggunakan JSON, terdapat dua hasil yakni *meta* dan *data*. Pada hasil *meta*, terdapat *key message* yang memberikan pesan bahwa akun telah berhasil mendapatkan akun pengguna, dan *key status* memberikan status sukses. Sedangkan, pada hasil *data*, memberikan informasi detail terkait pengguna yang telah sukses masuk. Informasi detail yang diberikan di antaranya adalah ID pengguna, nama, alamat surel, dan nomor telepon.

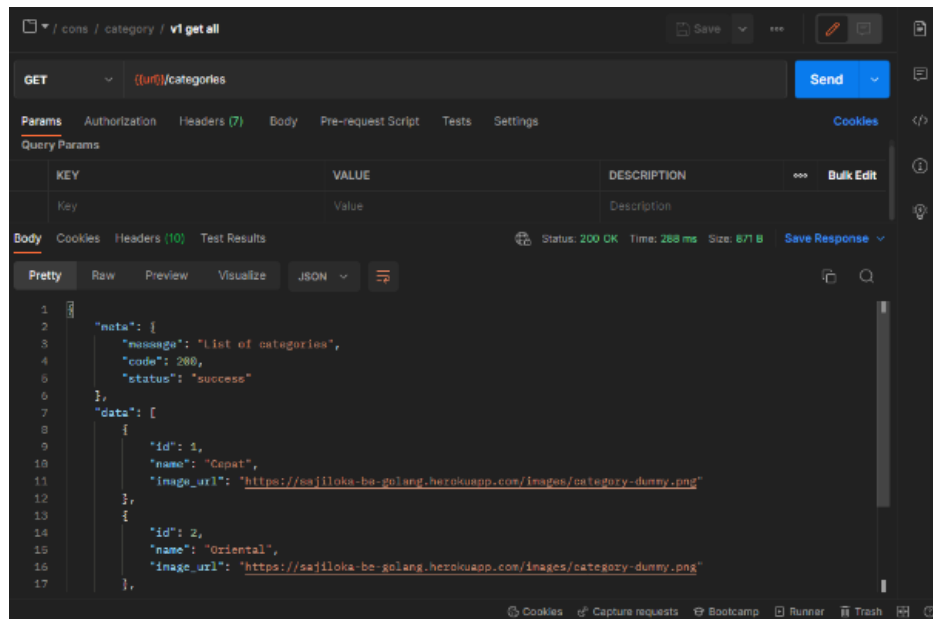


Gambar 4.3 API Dapatkan Akun Pengguna

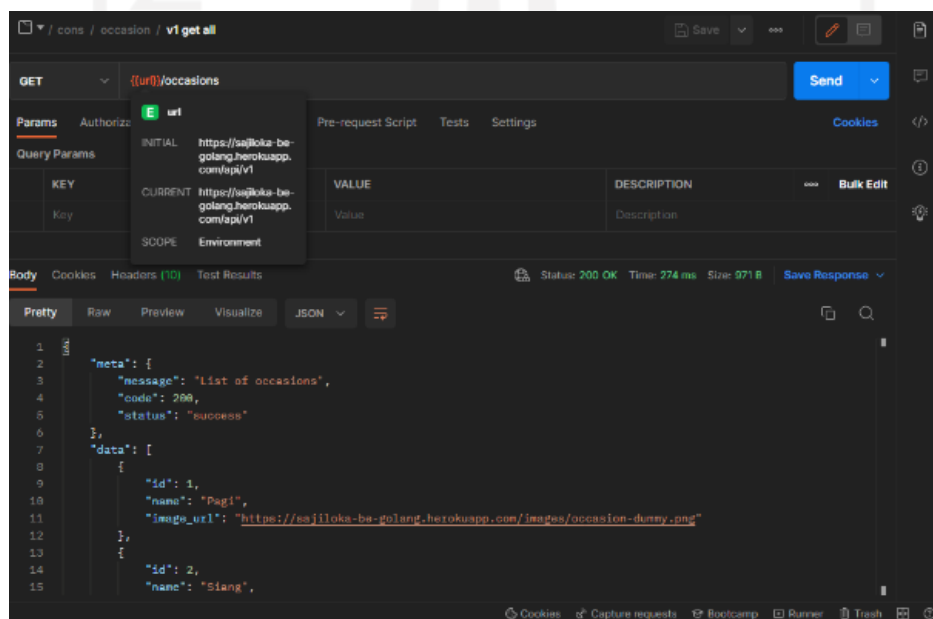
b. API Halaman Utama

Terdapat API yang mendukung kebutuhan data terhadap aplikasi *frontend* pada halaman utama, antara lain daftar kategori produk (*category*), daftar peristiwa produk (*occasion*), serta daftar menu produk. Ketiga API halaman utama ditunjukkan pada Gambar 4.4, Gambar 4.5, dan Gambar 4.6.

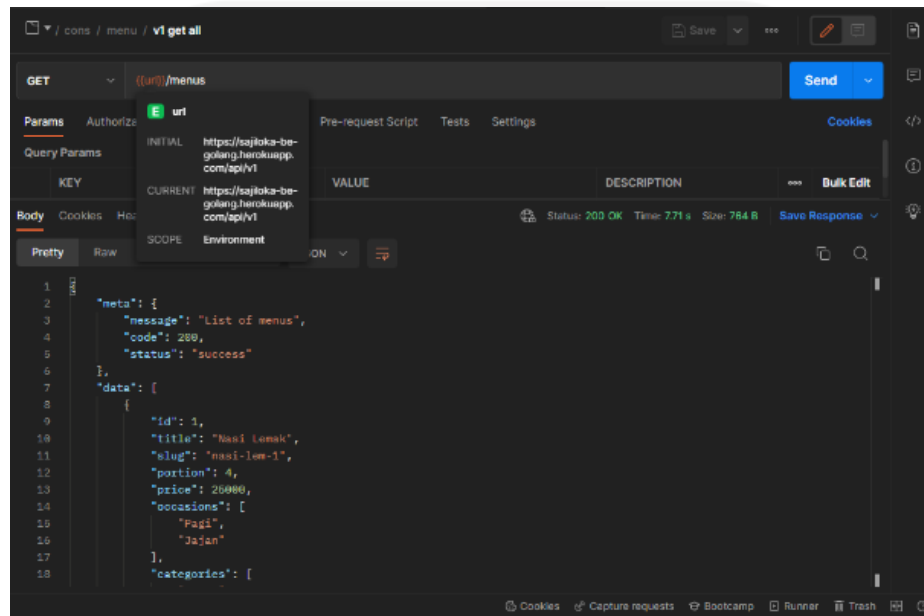
Halaman daftar kategori produk (*category*) yang ditunjukkan oleh Gambar 4.4 adalah salah satu kebutuhan data yang didukung oleh API Halaman Utama. Terdapat dua data hasil yang ditunjukkan dengan format JSON, yaitu *meta* dan *data*. Pada hasil *meta*, terdapat *key message* yang memberikan informasi mengenai daftar kategori. Sedangkan, pada hasil *data*, terdapat informasi detail mengenai kategori yang ada, seperti ID kategori, nama kategori, dan tautan gambar dari kategori yang dimaksud.

Gambar 4.4 API Daftar *Category*

Halaman daftar peristiwa produk (*occasion*) yang ditunjukkan oleh Gambar 4.5 adalah salah satu kebutuhan data yang didukung oleh API Halaman Utama. Terdapat dua data hasil yang ditunjukkan dengan format JSON, yaitu *meta* dan *data*. Pada hasil *meta*, terdapat *key message* yang memberikan informasi mengenai daftar peristiwa produk. Sedangkan, pada hasil *data*, terdapat informasi detail mengenai kategori yang ada, seperti ID kategori, nama kategori, dan tautan gambar dari kategori yang dimaksud.

Gambar 4.5 API Daftar *Occasion*

Halaman daftar menu produk (*menus*) yang ditunjukkan oleh Gambar 4.6 adalah salah satu kebutuhan data yang didukung oleh API Halaman Utama. Terdapat dua data hasil yang ditunjukkan dengan format JSON, yaitu *meta* dan *data*. Pada hasil *meta*, terdapat *key message* yang memberikan informasi mengenai daftar menu produk. Sedangkan, pada hasil *data*, terdapat informasi detail mengenai kategori yang ada, seperti ID produk, judul produk, *slug* produk, porsi produk, harga produk, dan *occasion* produk.

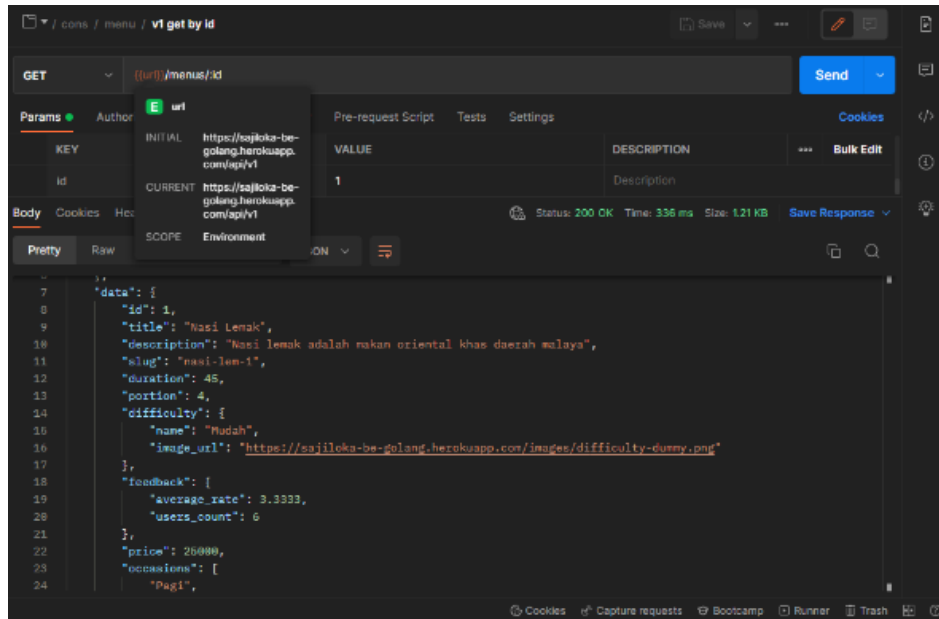


Gambar 4.6 API Daftar Produk

c. API Halaman Detail Produk

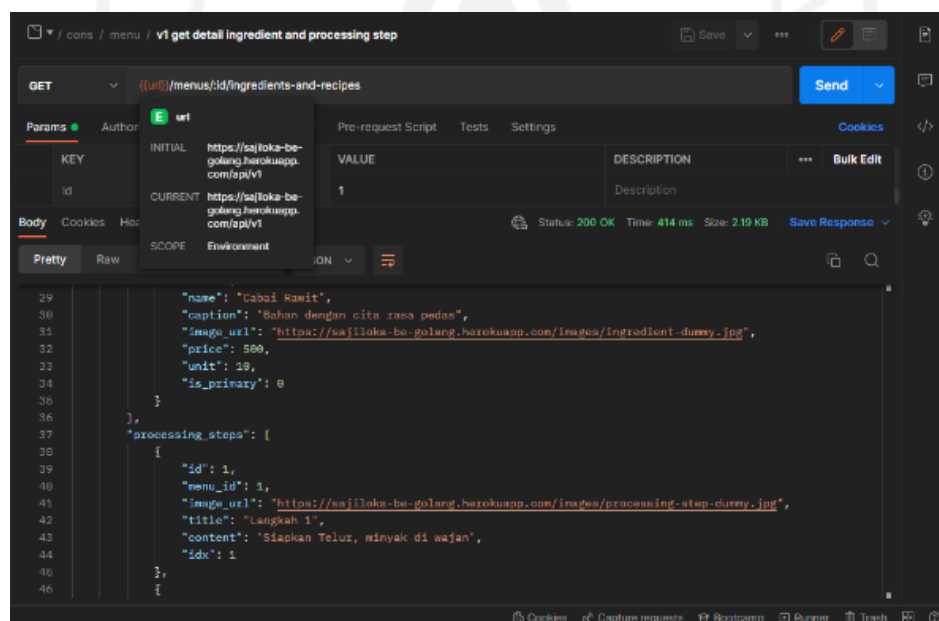
Terdapat API yang mendukung kebutuhan data terhadap aplikasi *frontend* pada halaman detail produk, di antaranya adalah halaman detail menu, dan halaman bahan dan cara memasak menu. Kedua API halaman detail produk ditunjukkan pada Gambar 4.7 dan Gambar 4.8.

Halaman daftar produk berdasarkan ID yang ditunjukkan oleh Gambar 4.7 adalah salah satu kebutuhan data yang didukung oleh API Detail Produk. Informasi detail mengenai produk yang disajikan menggunakan format JSON di antaranya adalah ID produk, judul produk, deskripsi produk, *slug* produk, durasi pembuatan produk, porsi produk, informasi detail mengenai tingkat kesulitan pembuatan produk, *feedback* dari pengguna berdasarkan *rating*, harga produk, dan *occasion* produk.



Gambar 4.7 API Detail Produk By ID

Halaman bahan dan cara memasak produk ditunjukkan oleh Gambar 4.8 adalah salah satu kebutuhan data yang didukung oleh API Detail Produk. Informasi detail mengenai produk yang disajikan menggunakan format JSON di antaranya adalah ID produk, nama produk, deskripsi produk, tauran gambar produk, harga produk, dan langkah memasak produk. Pada langkah memasak produk, terdapat urutan langkah yang diwakili oleh ID. Informasi detail mengenai langkah memasak produk juga diberikan secara detail agar mempermudah pengguna, seperti penjelasan (*content*) dan tautan gambar langkah memasak.

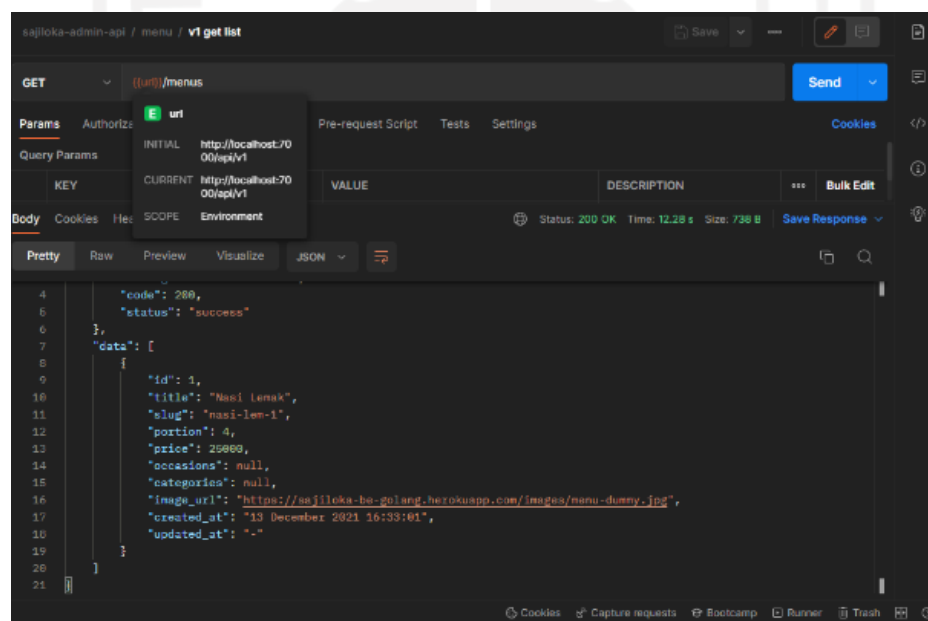


Gambar 4.8 API Bahan dan Cara Memasak Produk

d. API Halaman Daftar Produk (Admin)

Terdapat API yang mendukung kebutuhan data terhadap aplikasi *frontend* pada halaman daftar produk. API Halaman daftar produk tersebut ditunjukkan pada Gambar 4.9.

Halaman daftar produk ditunjukkan oleh Gambar 4.9 adalah salah satu kebutuhan data yang didukung oleh API Daftar Produk yang hanya dapat diakses dari sisi Admin. Informasi yang didapatkan dari API ini adalah informasi detail mengenai produk, seperti ID produk, judul produk, *slug* produk, porsi produk, harga produk, *occasion* produk, kategori produk, tautan gambar mengenai produk, waktu pembuatan produk, dan perubahan informasi produk. Hal-hal detail seperti waktu pembuatan produk dan perubahan informasi produk adalah hal yang hanya dapat diketahui oleh sisi admin saja.



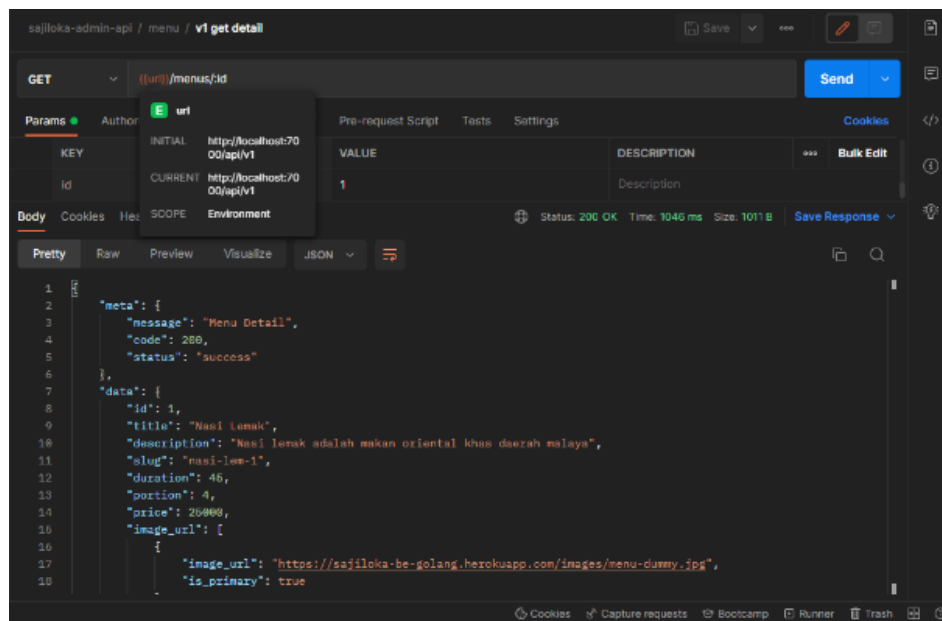
Gambar 4.9 API Daftar Produk (Admin)

e. API Halaman Detail Produk (Admin)

Terdapat API yang mendukung kebutuhan data terhadap aplikasi *frontend* pada halaman daftar produk. API Halaman daftar produk tersebut ditunjukkan pada Gambar 4.10.

Halaman detail produk ditunjukkan oleh Gambar 4.10 adalah salah satu kebutuhan data yang didukung oleh API Detail Produk yang hanya dapat diakses dari sisi Admin. Informasi yang didapatkan dari API ini adalah informasi detail mengenai produk, seperti ID produk, judul produk, deskripsi produk, *slug* produk, durasi pembuatan produk, porsi produk, harga produk, *occasion* produk, kategori produk, tautan gambar mengenai produk, waktu pembuatan produk,

dan perubahan informasi produk. Hal-hal detail seperti waktu pembuatan produk dan perubahan informasi produk adalah hal yang hanya dapat diketahui oleh sisi admin saja.

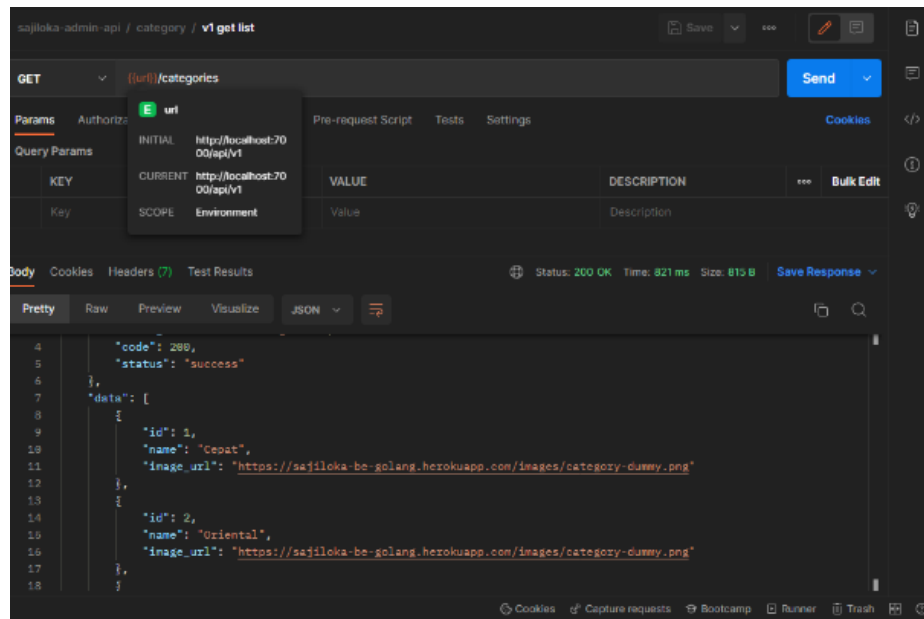


Gambar 4.10 API Detail Produk (Admin)

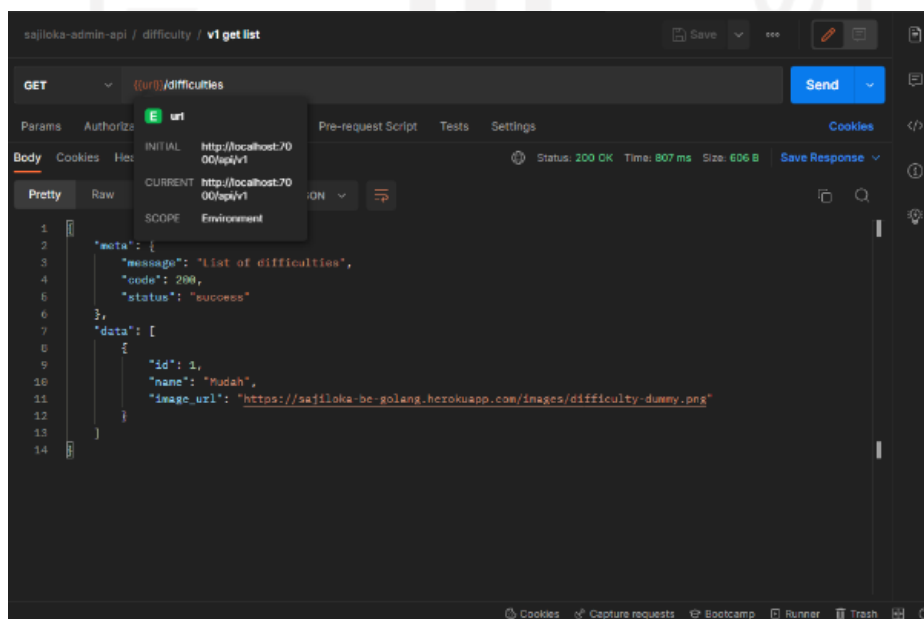
f. API Halaman Tambah Produk (Admin)

Terdapat API yang mendukung kebutuhan data terhadap aplikasi *frontend* pada halaman tambah produk, di antaranya adalah halaman daftar kategori produk, halaman daftar kategori kesulitan produk, dan halaman simpan produk. Ketiga API halaman tambah produk hanya dapat diakses dari sisi admin saja, serta ditunjukkan pada Gambar 4.11, Gambar 4.12, dan Gambar 4.13.

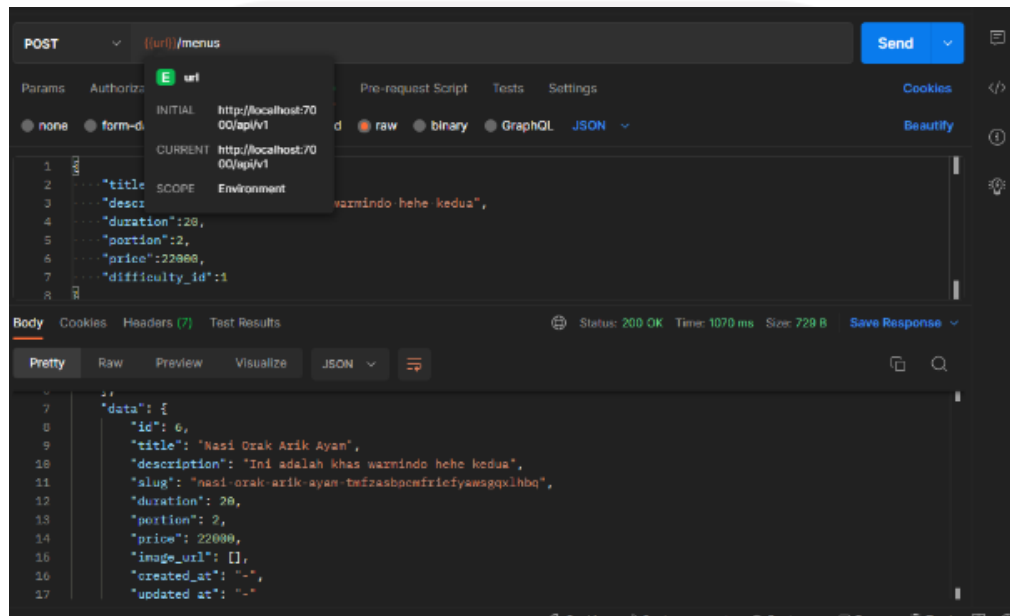
Halaman daftar kategori (*category*) yang ditunjukkan oleh Gambar 4.11 adalah salah satu kebutuhan data yang didukung oleh API Tambah Produk yang hanya dapat diakses dari sisi Admin. Informasi yang didapatkan dari API ini adalah informasi detail mengenai produk berdasarkan kategori, seperti ID kategori, nama kategori produk, dan tautan gambar kategori produk.

Gambar 4.11 API Daftar *Category* (Admin)

Halaman daftar kategori kesulitan produk (*difficulty*) yang ditunjukkan oleh Gambar 4.12 adalah salah satu kebutuhan data yang didukung oleh API Tambah Produk yang hanya dapat diakses dari sisi Admin. Informasi yang didapatkan dari API ini adalah informasi detail mengenai produk berdasarkan kategori, seperti ID *difficulty*, nama kategori kesulitan, dan tautan gambar kategori produk.

Gambar 4.12 API Daftar *Difficulty* (Admin)

Halaman daftar simpan data produk yang ditunjukkan oleh Gambar 4.13 adalah salah satu kebutuhan data yang didukung oleh API Tambah Produk yang hanya dapat diakses dari sisi Admin. Informasi yang didapatkan dari API ini adalah informasi detail mengenai produk berdasarkan kategori, seperti ID produk, judul produk, deskripsi produk, *slug* produk, durasi pembuatan produk, porsi produk, harga produk, tautan gambar mengenai produk, waktu pembuatan produk, dan waktu perubahan informasi terkait produk.



Gambar 4.13 API Simpan Data Produk (Admin)

4.2 Hasil Pengujian Aplikasi

Pada hasil pengujian ini semua proses *unit testing* mayoritas menggunakan *Mock Data* yang merupakan sebuah data palsu (*dummy*) yang merepresentasikan ekspektasi hasil keluaran dari fungsi yang dilakukan pengujian.

4.2.1 Sisi Pengguna (Consumer)

Berikut merupakan rekapitulasi dari pengujian aplikasi yang dilakukan oleh sisi pengguna pada aplikasi *backend* menggunakan *unit testing*, yang kemudian dikategorikan berdasarkan *service*, unit atau fungsi, dan hasil *unit testing*. Terdapat hasil pengujian dari enam *service* yang akan disajikan pada Tabel 4.1, yaitu: *Auth service*, *Menu service*, *Ingredient service*, *Category service*, *Occasion service*, dan *Difficulty service*.

Tabel 4.1 Hasil akhir *unit testing* Web API Pengguna

Hasil <i>Unit testing</i> Consumer API (Golang)			
<i>Service</i>	Unit	Result	Time
Auth	generateToken	PASS	0.509s
	validateToken	PASS	0.473s
Menu	getAllMenu	PASS	1.768s
	getMenuById	PASS	1.33s
	getProcessingStepsById	PASS	1.30s
Ingredient	getMenuIngredients	PASS	1.27s
Category	getAllCategories	PASS	11.44s
	getAllByMenuId	PASS	11.55s
Occasion	getAllOccasions	PASS	1.42s
	getAllByMenuId	PASS	1.29s
Difficulty	getAllDifficulties	PASS	9.56s
	getById	PASS	11.56s

4.2.2 Sisi Pengelola (Admin)

Berikut merupakan rekapitulasi dari pengujian aplikasi yang dilakukan oleh sisi pengelola atau admin pada Web API menggunakan *unit testing*, yang mana kemudian dikategorikan berdasarkan *service*, *unit* atau fungsi, dan hasil *unit testing*. Terdapat hasil pengujian dari satu *service* yang disajikan pada Tabel 4.2, yaitu *Menu service*.

Tabel 4.2 Hasil akhir *unit testing* Web API Admin

Hasil <i>Unit testing</i> Admin API (Golang)			
<i>Service</i>	Unit	Result	Time
Menu	createMenu	PASS	0.509s

Tabel 4.1 dan Tabel 4.2 menunjukkan bahwa service-service yang dilakukan pengujian menggunakan *unit test* telah lolos seratus persen, berarti aplikasi *backend* sudah berjalan dengan baik.

4.3 Refleksi

Seluruh kegiatan yang telah dilaksanakan diiringi dengan berbagai hambatan, tantangan, serta pencapaian. Guna memberikan pembelajaran terhadap kegiatan perintisan bisnis, khususnya kegiatan dalam pengembangan produk, maka akan disajikan semua temuan selama kegiatan berlangsung, mulai dari kendala, tantangan, dan pencapaian. Hasil dari pembelajaran tersebut kemudian dirangkum dalam wawasan kegiatan, sehingga pengalaman dari kegiatan yang dilakukan dapat bermanfaat bagi semua orang, terutama bagi *hacker* dari sebuah *startup*.

4.3.1 Kendala dan Hambatan

Selama berlangsungnya pengembangan Sajiloka, tidak dapat dipungkiri bahwa pasti terdapat kendala dan hambatan yang dihadapi selama kegiatan berlangsung. Kendala dan hambatan yang dihadapi di antaranya adalah waktu yang sempit, kurangnya komunikasi antar anggota, dan kurangnya jumlah anggota.

Waktu yang diberikan oleh panitia selama mengikuti kegiatan untuk memvalidasi masalah dan solusi ternyata terlampau cepat, sehingga solusi yang dihasilkan oleh *hipster* hanya dalam bentuk rancang bangun fitur aplikasi yang kasar dan tidak diuji dengan baik. Hal ini kemudian mengakibatkan adanya kesulitan dalam proses implementasi rancang bangun ke dalam aplikasi produksi. Kesulitan ini disebabkan karena fitur yang akan dibuat masih belum final, atau dengan kata lain, fitur belum diuji pada fase rancang bangun.

Kemudian, kurangnya koordinasi menjadi kendala selama proses pengembangan aplikasi berlangsung karena tidak diterapkan nya kegiatan *daily scrum* dengan disiplin. Tidak diterapkan nya kegiatan tersebut mengakibatkan proses berjalan nya *sprint* menjadi tidak teratur dan menjadi tidak sesuai dengan teori *Scrum* yang dijelaskan pada *Scrum Guide*.

Kendala terakhir yang dihadapi adalah kurangnya personil atau jumlah anggota. Kurangnya anggota yang membantu melaksanakan pengembangan aplikasi menjadi penyebab tidak sempurna nya aplikasi yang dihasilkan. Utamanya, pada pengembangan aplikasi terdapat anggota yang berperan sebagai manajer proyek (*scrum master*), yang mana berperan dalam mengawasi dan memanajemeni jalannya kegiatan pengembangan aplikasi.

Pada akhirnya, tim Sajiloka tidak dapat melanjutkan ke tahap inkubasi atau *Hatch*, karena aplikasi yang telah dikembangkan tidak mampu dirilis. Aplikasi dirasa belum siap untuk menangani transaksi dari pengguna karena fitur-fitur penting, seperti “Halaman Keranjang” dan “Halaman *Checkout*” tidak masuk ke dalam *sprint*, karena belum terdapat rancang bangun halaman tersebut dari Hipster. Faktor lain yang menyertai adalah aplikasi sisi admin yang

berguna untuk pengelolaan data produk serta untuk memproses transaksi pengguna tidak selesai dikembangkan ketika sudah berada pada tenggat waktu pelaksanaan kegiatan *Bootcamp*. Dengan berakhirnya perjalanan tim Sajiloka pada kegiatan Gerakan 1000 *Startup* Digital, proses pengembangan aplikasi tidak dapat dilanjutkan lagi. Terdapat penyesalan yang dirasakan karena komposisi tim Sajiloka telah terdiri dari *hustler*, *hispster*, dan *hacker* serta telah memiliki kemampuan yang cukup untuk dapat mengembangkan sebuah aplikasi.

4.3.2 Tantangan

Selain adanya kendala dan hambatan, tantangan juga dihadapi oleh tim Sajiloka dalam kegiatan pengembangan aplikasi berlangsung. Terdapat sedikitnya dua tantangan yang dihadapi, antara lain adalah panitia yang memberikan waktu yang relatif sedikit untuk mengembangkan aplikasi, dan anggota dari tim pengembang yang hanya terdiri dari dua orang saja.

Tantangan pertama yang dihadapi selama pengembangan aplikasi berlangsung adalah panitia yang memberikan waktu yang relative sedikit. Hal ini menjadi sebuah tantangan karena dalam waktu satu bulan, yang mana waktu tersebut relatif sedikit, tim pengembang diminta untuk dapat mengembangkan aplikasi tahap MVP (*Most Viable Product*).

Kemudian, tantangan dalam pengembangan aplikasi bertambah karena kurangnya anggota. Sayangnya, tim pengembang hanya terdiri dari dua orang anggota, yang mana perlu perjuangan untuk dapat menyelesaikan pengembangan aplikasi hingga tahap MVP dalam waktu yang sempit.

4.3.3 Capaian

Setelah menjelaskan terkait kendala dan tantangan yang dihadapi, maka terdapat pencapaian yang dicapai selama pengembangan aplikasi dilaksanakan. Capaian yang diraih di antaranya adalah aplikasi pada sisi *backend* telah berhasil di-*deploy* pada jasa *server* demo di Heroku.com, sedangkan aplikasi pada sisi *frontend* Android yang mencakup fitur-fitur utama telah berhasil dikembangkan dan terintegrasi dengan *service backend* yang berada pada *server* demo di Heroku.com.

4.3.4 Wawasan Kegiatan

Hasil pembelajaran dari pengembangan aplikasi yang berlangsung kemudian dirangkum pada sub-bab wawasan kegiatan. Wawasan kegiatan yang dijabarkan diharapkan dapat bermanfaat bagi semua orang, terutama bagi *hacker* dari sebuah *startup*. Beberapa wawasan

kegiatan yang didapatkan adalah komunikasi menjadi kunci utama, dan fitur harus didefinisikan dengan jelas sesuai dengan kebutuhan.

Komunikasi menjadi kunci utama dalam kegiatan pengembangan aplikasi. Pada penerapan *Scrum* telah diberikan kegiatan yang merupakan implementasi junjungan nilai komunikasi, seperti *Sprint Planning*, *Daily Scrum*, dan *Sprint Retrospective*. Tidak berjalan nya ketiga kegiatan tersebut dengan disiplin membuat kinerja tim menjadi rawan miskomunikasi dan menjadi jalannya kegiatan tidak teratur.

Kemudian, semua fitur harus didefinisikan dengan jelas melalui kebutuhan fitur (*feature requirement*) sehingga hasil yang dikerjakan oleh pengembang bisa sesuai dengan kebutuhan dari *stakeholder* (*product owner* hingga pengguna). Pada umumnya pendefinisian fitur menjadi tanggung jawab tim produk (*hipster*) karena pengembang (*hacker*) hanya bertugas untuk merealisasikan fitur dalam bentuk desain kedalam aplikasi produksi.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Semua kegiatan pengembangan aplikasi telah selesai dilaksanakan dan menghasilkan beberapa hasil yang telah dipaparkan pada bab sebelumnya. Berikut merupakan kesimpulan dari kegiatan pengembangan aplikasi yang telah dilaksanakan.

- a. *Task-task* pengembangan aplikasi *backend* yang telah diselesaikan sehingga menghasilkan kumpulan API dan telah berhasil lolos uji *unit testing* menunjukkan bahwa aplikasi backend memiliki kualitas baik serta mampu mendukung kebutuhan data pada aplikasi sisi *frontend*.
- b. Tidak semua *task* yang diberikan mampu untuk diselesaikan karena waktu *sprint* telah habis dan kegiatan perintisan bisnis 1000SD telah berakhir sehingga aplikasi Sajiloka tidak dapat dirilis ke tahap selanjutnya, tahap produksi. Salah satu faktor lain kegagalan dalam menyelesaikan *task* adalah karena desain yang diberikan belum final (tidak dilakukan pengujian desain oleh tim perancang desain atau *hispter*).
- c. Hasil akhir dari kegiatan 1000SD adalah startup Sajiloka dinyatakan tidak lolos ke tahap inkubasi atau *hatch*.

5.2 Saran

Melalui refleksi dan kesimpulan dari kegiatan yang telah dilaksanakan terdapat beberapa poin yang bisa disampaikan sebagai saran dalam pengembangan-pengembangan aplikasi selanjutnya yaitu sebagai berikut:

- a. Definisi *task* secara jelas dan lengkap merupakan kunci awal dari kelancaran sebuah proyek, sehingga sebelum dimulainya kegiatan pengembangan diharapkan semua *task* yang akan dikerjakan sudah memenuhi kualifikasi kesiapan *task* atau biasa disebut *definition of ready* (DoR). Jika sebuah *task* tidak memenuhi kriteria tersebut, maka *task* tersebut tidak layak untuk masuk ke dalam tahap pengembangan. Jika dipaksakan untuk masuk ke dalam tahap pengembangan, akan menyebabkan pengembang kebingungan dan berakibat pada ketidaklancaran dalam kegiatan pengembangan.
- b. Sebuah aplikasi yang siap produksi harus terus dilakukan iterasi *sprint* untuk improvisasi fitur, sehingga aplikasi tetap sesuai dengan kebutuhan pengguna dengan seiring nya waktu.

DAFTAR PUSTAKA

- 1000SD. (n.d.). *Beranda - 1000 Startup Digital*. Retrieved from <https://1000startupdigital.id/>
- Bai, Y. (n.d.). *Best practices: Why use Golang for your project*. Retrieved from <https://uptech.team/blog/why-use-golang-for-your-project>
- Coursera. (2022). *What Does a Back-End Developer Do? | Coursera*. Retrieved from <https://www.coursera.org/articles/back-end-developer>
- GeeksforGeeks. (2020). *SQL vs NoSQL: Which one is better to use? - GeeksforGeeks*. Retrieved from <https://www.geeksforgeeks.org/sql-vs-nosql-which-one-is-better-to-use/?ref=lbp>
- IBM Cloud Education. (2021). *What is a REST API? | IBM*. Retrieved from <https://www.ibm.com/cloud/learn/rest-apis>
- Mackinnon, T., Freeman, S., & Craig, P. (2001). Endo-Testing : Unit Testing with Mock Objects. *Extreme Programming Examined*, 287–301. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.3214&rep=rep1&type=pdf>
- Nidhra, S., & Dondeti, J. (2012). BLACK BOX AND WHITE BOX TESTING TECHNIQUES –A LITERATURE REVIEW. *Project Management Journal*, 2(2), 29–50.
- Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., & Abrahamsson, P. (2014). Software development in startup companies: A systematic mapping study. *Information and Software Technology*, 56(10), 1200–1218. <https://doi.org/10.1016/j.infsof.2014.04.014>
- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. (November).
- Singhal, P. (2021). Frontend vs Backend - GeeksforGeeks. *GeeksforGeeks*. Retrieved from <https://www.geeksforgeeks.org/frontend-vs-backend/>
- Stylos, J., Faulring, A., Yang, Z., & Myers, B. A. (2009). Improving API documentation using API usage information. *2009 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2009*, 119–126. <https://doi.org/10.1109/VLHCC.2009.5295283>
- Thomas, M. (2014). *A Perfect Startup Team: The Golden Triangle*. Retrieved from <https://kmvidxgosnelleb.quora.com/A-Perfect-Startup-Team-The-Golden-Triangle>

LAMPIRAN

