

**PENGEMBANGAN BACK END APLIKASI SISTEM
INFORMASI TERPADU BSI PINTER
MENGUNAKAN
SPRING FRAMEWORK**



Disusun Oleh:

N a m a : Muhammad Devano Zaidan
NIM : 20523034

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

2024

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**PENGEMBANGAN BACKEND APLIKASI SISTEM
INFORMASI TERPADU BSI PINTER MENGGUNAKAN
SPRING FRAMEWORK**

TUGAS AKHIR JALUR MAGANG



N a m a
NIM

: Muhammad Devano Zaidan
: 20523034

Yogyakarta, 27 Oktober 2024

Pembimbing,



(Andhik Budhi Cahyono, S.T, M.T)

HALAMAN PENGESAHAN DOSEN PENGUJI

**PENGEMBANGAN BACKEND APLIKASI SISTEM
INFORMASI TERPADU BSI PINTER MENGGUNAKAN
SPRING FRAMEWORK**

TUGAS AKHIR JALUR MAGANG

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, 27 Oktober 2024

Tim Penguji

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

**Anggota 1**

Sheila Nurul Huda, S.Kom., M.Cs.

**Anggota 2**

Mukhammad Andri Setiawan, Ph.D.



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 : Muhammad Devano Zaidan
NIM : 20523034

Tugas akhir dengan judul:

PENGEMBANGAN BACKEND APLIKASI SISTEM INFORMASI TERPADU BSI PINTER MENGGUNAKAN SPRING FRAMEWORK

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, 27 Oktober 2024



(Muhammad Devano Zaidan)

HALAMAN PERSEMBAHAN

Puji syukur penulis panjatkan kehadiran Allah SWT, atas segala nikmat dan karunia yang telah diberikan dalam menyelesaikan tugas akhir ini yang berjudul “Pengembangan Back End Aplikasi Sistem Informasi Terpadu BSI PINTER Menggunakan Spring Framework”, sebagai salah satu syarat untuk mendapatkan gelar kesarjanaan. Penulis tidak ingin menampik bahwa skripsi ini masih jauh dari kata sempurna, tetapi penulis sangat bangga telah mencapai titik ini, yang akhirnya skripsi bisa diselesaikan di waktu yang tepat.

Napoleon Hill pernah berkata “*What the mind of man can conceive and believe, it can achieve.*” Kutipan dari beliau menginspirasi penulis untuk terus berjuang dan percaya bahwa setiap impian dapat terwujud dengan kerja keras dan keyakinan. Skripsi ini merupakan manifestasi dari keyakinan dan usaha yang sungguh-sungguh penulis untuk menghasilkan sebuah karya yang berharga dan dapat bermanfaat bagi perkembangan ilmu pengetahuan.

Skripsi atau tugas akhir ini saya persembahkan untuk:

- Ayah dan ibu, Warno dan Yuliana Dhewi terimakasih atas doa, semangat, motivasi, pengorbanan, nasehat serta kasih sayang yang telah diberikan tanpa henti sampai saat ini.
- Adikku tersayang, Reyhan Kayko Izzaki yang merupakan teman hidup saya sedari lahir sampai kapanpun, yang telah menemani saya dan menjadi penyemangat dalam menyelesaikan tugas akhir ini.
- Semua teman-teman Informatika yang telah mengenal saya selama masa studi, khususnya angkatan 2020.
- Kepada semua teman-teman, saudara yang tidak bisa saya sebutkan satu per satu, saya persembahkan tugas akhir ini untuk kalian semua.

HALAMAN MOTO

“Orang-orang yang mudah menyerah tidak akan menjadi pemenang dan pemenang tidak akan pernah menyerah”

KATA PENGANTAR

Alhamdulillah, dengan rahmat Allah SWT Yang Maha Pengasih dan Maha Penyayang, puji syukur penulis panjatkan kehadiran-Nya yang telah memberikan anugerah kasih sayang, rezeki, kesehatan, dan berkah serta atas ridho dan hidayah-Nya penulis dapat menyelesaikan skripsi dengan judul “Pengembangan Back End Aplikasi Sistem Informasi Terpadu BSI PINTER Menggunakan Framework Spring”. Shalawat dan salam penulis panjatkan pada Nabi Muhammad SAW yang mengantarkan kita Zaman yang gelap menuju Zaman yang terang benderang seperti saat ini, serta telah menjadi suri teladan bagi umat islam dalam menjalani perintah dan menjauhi larangan-Nya.

Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar sarjana sekaligus sebagai tanggungjawab akhir penulis sebagai mahasiswa jurusan Informatika Universitas Islam Indonesia. Penulis sadar bahwa dalam penyusunan tugas akhir ini masih banyak kesalahan dan kekurangan, oleh karena itu penulis mengharapkan saran dan kritikan dari berbagai pihak untuk menjadi bahan evaluasi penulis dalam usaha melakukan perbaikan dan penyempurnaan penulisan tugas akhir ini.

Dalam penyelesaian studi dan skripsi ini, penulis banyak memperoleh bantuan baik pengajaran, bimbingan, arahan dan motivasi dari berbagai pihak baik secara langsung maupun tidak langsung. Untuk itu, penulis ingin menyampaikan penghargaan dan ucapan terima kasih sebanyak-banyaknya kepada:

1. Kedua orang tua Ayah Warno dan Ibu Yuliana Dhewi, yang selalu memberikan dukungan penuh kepada penulis berupa doa, kehadiran yang sangat berarti, motivasi dan semangat yang diberikan tanpa henti yang membuat penulis tetap bertahan hingga saat ini.
2. Bapak Andhik Budhi Cahyono, selaku dosen pembimbing skripsi yang telah memberikan arahan penulisan laporan ini.
3. PT Bank Syariah Indonesia, yang telah memberikan kesempatan kepada penulis untuk melaksanakan magang dan merasakan langsung lingkungan kerja pada lingkup IT perbankan syariah.
4. Mentor-mentor yang bekerja di Bank Syariah Indonesia khususnya di bagian departemen pengembangan IT yang telah membagikan pengetahuan dan pengalamannya selama pelaksanaan magang, serta memberikan arahan selama pengembangan proyek.

5. Serta seluruh pihak yang yang tidak bisa penulis sebutkan satu per satu, penulis mengucapkan rasa terimakasih banyak atas doa dan dukungannya serta mohon maaf yang sebesar-besarnya. Semoga segala kebaikan, bantuan dan amal baik dari berbagai pihak tersebut mendapat balasan yang setimpal dari Allah SWT. Penulis berharap semoga skripsi yang dibuat ini dapat bermanfaat bagi semua pihak. *Aaamiin..*

Yogyakarta, 27 Oktober 2024

A handwritten signature in black ink, featuring a stylized, cursive script that appears to read 'Muhammad Devano Zaidan'.

(Muhammad Devano Zaidan)

SARI

Perkembangan teknologi menuntut seluruh instansi, termasuk sektor perbankan, untuk secara adaptif mengadopsi teknologi demi meningkatkan inovasi produk mereka sehingga mampu memberikan pelayanan terbaik kepada nasabah. Divisi pengembangan IT BSI, IDG (IT Development Group) berencana mengembangkan sebuah solusi digital dalam rangka meningkatkan aksesibilitas informasi dan evaluasi pemahaman karyawan terhadap produk-produk BSI. Solusi tersebut direalisasikan dengan mengembangkan aplikasi BSI PINTER. BSI PINTER merupakan sebuah aplikasi Sistem Informasi Terpadu yang menyediakan akses informasi terpusat untuk karyawan serta kebutuhan operasional, informasi produk BSI, penilaian kuis untuk karyawan, dan pengelolaan ruang rapat. Pada makalah ini disajikan pengembangan aplikasi *backend* dengan lingkup pada fitur kuis sebagai sarana evaluasi pemahaman karyawan dan pengelolaan ruang *meeting*. Aplikasi dikembangkan dengan menggunakan metodologi agile menggunakan metode Scrum. Aplikasi dikembangkan dengan menggunakan framework Spring dengan dukungan Spring Boot dengan menggunakan arsitektur REST API. Spring framework dipilih karena kemampuannya dalam menyediakan struktur aplikasi yang solid, modul keamanan yang teruji, kemudahan dalam pengembangan dan pengelolaan aplikasi, serta mendukung arsitektur REST API yang memudahkan integrasi dan skalabilitas aplikasi. Hasil dari pengembangan ini adalah API fitur yang merupakan sebuah hasil dari eksekusi Sprint yang telah dilaksanakan pada kegiatan magang penulis selama di PT Bank Syariah Indonesia. Fitur yang telah dikembangkan kemudian dilakukan pengujian dan peninjauan ulang, setiap hasil implementasi ditinjau berdasarkan kriteria penerimaan pada setiap *user story*.

Kata kunci: Spring, SpringBoot, REST API, *Backend*, Scrum.

GLOSARIUM

Runtime	fase ketika program yang sudah dikompilasi dijalankan dan dieksekusi oleh sistem operasi, termasuk pengelolaan memori dan interaksi dengan perangkat keras.
Debug	langkah untuk menelusuri kesalahan kode program.
Framework	kerangka kerja yang menyediakan struktur dasar dan alat untuk mempermudah pengembangan aplikasi perangkat lunak.
Cloud Storage	metode penyimpanan data secara digital di server jarak jauh yang dapat diakses melalui internet, memungkinkan pengelolaan dan berbagi data secara fleksibel dan aman.
Hibernate	framework Java yang digunakan untuk memetakan objek di aplikasi ke tabel <i>email</i> , memudahkan operasi <i>email</i> dengan mengelola penyimpanan, pengambilan, dan pembaruan data secara otomatis

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	ix
GLOSARIUM	x
DAFTAR ISI	xi
DAFTAR TABEL	xii
DAFTAR GAMBAR	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Ruang Lingkup	2
1.3 Tujuan	3
1.4 Manfaat	3
1.5 Sistematika Penulisan	4
BAB II LANDASAN TEORI	6
2.1 <i>Backend</i>	6
2.2 REST API	6
2.3 Spring Framework	8
2.4 Spring Boot	10
2.5 Spring Security	11
2.6 Spring Data JPA	12
2.7 Scrum	14
BAB III PELAKSANAAN MAGANG	17
3.1 Aktivitas Magang	17
3.2 Manajemen Proyek	18
3.3 Pelaksanaan Proyek BSI PINTER	28
3.4 Hasil Pengembangan Aplikasi	88
3.5 Status Proyek	90
BAB IV REFLEKSI PELAKSANAAN MAGANG	91
4.1 Relevansi Akademik	91
4.2 Pembelajaran Magang	102
BAB V PENUTUP	104
5.1 Kesimpulan	104
5.2 Saran	104
DAFTAR PUSTAKA	106
LAMPIRAN	108

DAFTAR TABEL

Tabel 3. 1 Backlog fitur autentikasi dan regsitrasi	30
Tabel 3. 2 Backlog fitur kuis <i>user</i>	32
Tabel 3. 3 Backlog fitur kuis admin.....	36
Tabel 3. 4 Backlog fitur pemesanan ruang <i>meeting</i> sisi <i>user</i>	39
Tabel 3.4 REST URL autentikasi dan registrasi	88
Tabel 3.5 REST URL fitur kuis	89
Tabel 3.6 REST URL fitur pengelolaan ruang <i>meeting</i>	89

DAFTAR GAMBAR

Gambar 2.1 Spring framework Runtime.....	9
Gambar 2.2 Spring Security Filter Chain	12
Gambar 2.3 Diagram Lapisan Aplikasi Spring Data JPA	13
Gambar 3.1 Scrum Lifecycle	19
Gambar 3.2 Bagian <i>feedback</i> Sprint Retrospective Meeting	23
Gambar 3.3 Timeline Jira	25
Gambar 3.4 Dependency antar item pada timeline	25
Gambar 3.5 Backlog Jira pada BSI PINTER.....	26
Gambar 3.6 Jira Board	27
Gambar 3.7 Confluence pada proyek BSI PINTER	28
Gambar 3.8 Halaman Autentikasi.....	31
Gambar 3.9 Halaman Verifikasi OTP.....	31
Gambar 3. 10 Halaman pendaftaran akun.....	32
Gambar 3. 11 Halaman product knowledge	33
Gambar 3. 12 Halaman detail kuis.....	34
Gambar 3. 13 Halaman hasil kuis	35
Gambar 3. 14 Halaman membagi hasil kuis kedalam <i>feeds</i>	36
Gambar 3. 15 Halaman utama admin	36
Gambar 3. 16 Halaman detail product knowledge admin.....	37
Gambar 3. 17 Halaman admin untuk menambahkan kuis	37
Gambar 3. 18 Bottom sheets menambahkan judul kuis.....	38
Gambar 3. 19 Alur halaman menambahkan soal kuis pada admin.....	38
Gambar 3. 20 Halaman utama <i>request</i> ruang <i>meeting</i>	40
Gambar 3. 21 Halaman detail ruang <i>meeting</i>	40
Gambar 3. 22 Alur halaman pemesanan ruang <i>meeting</i>	41
Gambar 3. 23 Halaman riwayat pemesanan ruang <i>meeting</i>	41
Gambar 3. 24 Arsitektur Spring yang digunakan di BSI PINTER	42
Gambar 3. 25 Alur pengembangan <i>backend</i>	45
Gambar 3. 26 ERD untuk autentikasi dan registrasi.....	46
Gambar 3. 27 Alur registrasi.....	46
Gambar 3. 28 Sequence diagram alur registrasi <i>backend</i>	47
Gambar 3. 29 Request response generate OTP	47

Gambar 3. 30 Alur pemuatan kode OTP	48
Gambar 3. 31 Request response verifikasi OTP	49
Gambar 3. 32 Alur verifikasi OTP.....	49
Gambar 3. 33 Request response pendaftaran akun	50
Gambar 3. 34 Halaman <i>login</i>	50
Gambar 3. 35 Request Response <i>login</i>	51
Gambar 3. 36 Alur pemrosesan <i>login</i>	51
Gambar 3. 37 Claims JWT pada BSI PINTER.....	52
Gambar 3. 38 Proses validasi kredensial <i>user</i>	52
Gambar 3. 39 Alur request pada Spring Security	53
Gambar 3. 40 Class JwtAuthenticationFilter	53
Gambar 3. 41 Alur proses validasi JWT	54
Gambar 3. 42 Konfigurasi Spring Security.....	55
Gambar 3. 43 Alur pengerjaan kuis pada <i>user</i>	56
Gambar 3. 44 ERD kuis	57
Gambar 3. 45 Halaman product knowledge	58
Gambar 3. 46 <i>Request</i> Contract mendapatkan daftar kuis	59
Gambar 3. 47 Flowchart mengambil daftar kuis pada product knowledge tertentu	59
Gambar 3. 48 Halaman detail kuis.....	60
Gambar 3. 49 <i>Request</i> contract detail kuis.....	61
Gambar 3. 50 Flowchart mengambil detail kuis untuk dijawab oleh insan BSI.....	62
Gambar 3. 51 <i>Request</i> contract menjawab soal kuis.....	63
Gambar 3. 52 Flowchart menjawab soal pada kuis	63
Gambar 3. 53 Halaman hasil akhir kuis.....	64
Gambar 3. 54 <i>Request</i> contract mendapatkan hasil akhir kuis	65
Gambar 3. 55 Flowchart memproses hasil akhir kuis.....	65
Gambar 3. 56 Halaman membagikan hasil kuis	66
Gambar 3. 57 <i>Request</i> contract membagikan hasil kuis	67
Gambar 3. 58 Flowchart membagikan hasil kuis.....	67
Gambar 3. 59 Halaman menambahkan judul pada kuis	68
Gambar 3. 60 <i>Request</i> contract menambahkan judul pada kuis	69
Gambar 3. 61 Sequence diagram menambahkan judul pada kuis	69
Gambar 3. 62 Alur halaman menambahkan pertanyaan/soal pada kuis	70
Gambar 3. 63 <i>Request</i> contract menambahkan pertanyaan pada kuis	71

Gambar 3. 64 Sequence diagram menambahkan pertanyaan pada kuis	71
Gambar 3. 65 Alur halaman mengubah data kuis.....	72
Gambar 3. 66 <i>Request</i> contract memperbarui judul kuis.....	72
Gambar 3. 67 <i>Request</i> contract memperbarui pertanyaan pada kuis	73
Gambar 3. 68 Sequence diagram memperbarui data kuis.....	74
Gambar 3. 69 Alur halaman menghapus kuis.....	74
Gambar 3. 70 <i>Request</i> contract menghapus kuis	75
Gambar 3. 71 Sequence diagram menghapus kuis	75
Gambar 3. 72 Alur halaman pemesanan ruang <i>meeting</i>	75
Gambar 3. 73 ERD fitur pemesanan ruang <i>meeting</i>	76
Gambar 3. 74 Halaman utama fitur pemesanan ruang <i>meeting</i>	77
Gambar 3. 75 Contract <i>request</i> mendapatkan daftar ruang <i>meeting</i>	78
Gambar 3. 76 Sequence diagram mendapatkan daftar ruang <i>meeting</i>	78
Gambar 3. 77 Halaman detail ruang <i>meeting</i>	79
Gambar 3. 78 <i>Request</i> contract mendapatkan detail ruang <i>meeting</i>	80
Gambar 3. 79 Sequence diagram mendapatkan detail ruang <i>meeting</i>	80
Gambar 3. 80 Alur halaman pemesanan ruang <i>meeting</i>	81
Gambar 3. 81 <i>Request</i> contract pemesanan ruang <i>meeting</i>	81
Gambar 3. 82 Sequence diagram pemesanan ruang <i>meeting</i>	82
Gambar 3. 83 Alur halaman riwayat pemesanan ruang <i>meeting</i>	83
Gambar 3. 84 <i>Request</i> contract daftar riwayat pemesanan ruang <i>meeting</i>	84
Gambar 3. 85 Sequence diagram mendapatkan riwayat pemesanan ruang <i>meeting</i>	84
Gambar 3. 86 Alur kerja proses penyimpanan file pada BSI PINTER	85
Gambar 3. 87 Dependensi Spring Cloud GCP pada Spring	86
Gambar 3. 88 Contoh konfigurasi Google Cloud Platform pada BSI PINTER	86
Gambar 3. 89 Alur proese penambahan dan penghapusan objek di Google Cloud Storage ...	87
Gambar 4. 1 Dependency Injection pada KantorCabangService.....	92
Gambar 4. 2 Contoh <i>circular dependency</i>	93
Gambar 4. 3 Kueri JPA menggunakan JPQL	96
Gambar 4. 4 Entitas MeetingRoom menggunakan metode <i>fetch</i> Lazy	97
Gambar 4. 5 Profilling aplikasi Spring menggunakan IntelliJIdea	97

BAB I

PENDAHULUAN

1.1 Latar Belakang

PT Bank Syariah Indonesia merupakan sebuah perusahaan BUMN yang bergerak di bidang perbankan syariah. BSI memiliki sebuah departemen atau unit kerja pada bidang pengembangan IT yang bernama IT Development Group. IDG memiliki tanggung jawab atas pengembangan solusi digital untuk pemenuhan kebutuhan internal dan eksternal perusahaan pada bidang IT. Selain itu, IDG juga memiliki tanggungjawab untuk mengelola manajemen informasi terhadap karyawannya. IDG memberikan informasi mengenai produk-produk BSI secara rutin kepada karyawan, salah satunya melalui SPB (Standar Prosedur Bisnis). SPB berisi informasi produk bisnis BSI yang diperbaharui secara berkala. Mengenal produk-produk bisnis BSI merupakan kewajiban bagi karyawan IDG dan seluruh insan BSI.

Materi produk yang diberikan kepada insan BSI tidak sebatas disampaikan saja, tetapi juga dilakukan evaluasi pemahaman karyawan terhadap materi produk yang sudah disampaikan. Salah satu metode evaluasi yang digunakan adalah dengan melaksanakan penilaian dan pengujian pemahaman melalui pengerjaan kuis. Kuis diberikan melalui Google Form dan disampaikan setelah penyampaian materi produk BSI. Hasil kuis dijadikan bahan evaluasi oleh departemen untuk menilai pemahaman produk insan BSI.

Meskipun metode pengerjaan kuis telah memberikan kemudahan dalam pengukuran pemahaman terhadap materi produk BSI, terdapat beberapa kendala yang dihadapi dalam pelaksanaannya. Salah satu masalah utama adalah keterbatasan fitur mengenai pelacakan kemajuan dan analisis hasil secara mendalam. Google Form memang menyediakan data hasil kuis dalam bentuk *spreadsheet*, tetapi diperlukan waktu dan usaha ekstra untuk melakukan analisis secara komprehensif. Pencatatan kemajuan hasil evaluasi karyawan juga kurang terdata dengan baik. Penanggungjawab harus mengumpulkan beberapa data hasil kuis untuk dianalisa. Oleh karena itu, diperlukan sebuah solusi untuk mengembangkan sebuah fitur yang secara efektif dapat melakukan pencatatan, pelacakan kemajuan, dan analisa hasil evaluasi dari pengerjaan kuis yang dilakukan oleh insan BSI.

Berdasarkan permasalahan tersebut, diperlukan sebuah solusi yaitu sebuah sistem terintegrasi yang mengintegrasikan kebutuhan pada manajemen informasi dan kebutuhan operasional perusahaan. Sistem ini diharapkan mampu memfasilitasi pencatatan, pelacakan kemajuan, dan hasil analisis evaluasi secara efektif dan efisien, serta menyediakan fitur yang

transparan mengenai pengelolaan ruang *meeting*. Sistem tersebut kemudian direalisasikan dalam bentuk sebuah aplikasi sistem informasi terpadu bernama BSI PINTER. Harapannya aplikasi tersebut dapat menyelesaikan permasalahan yang dialami dengan menyediakan solusi yang terintegrasi sehingga memudahkan akses informasi dan kebutuhan operasional karyawan.

Laporan ini bertujuan untuk menyajikan hasil implementasi penggunaan framework Spring untuk mengembangkan *backend* untuk sistem informasi meliputi fitur kuis dan pengelolaan ruang meeting pada aplikasi BSI PINTER. Dengan memanfaatkan arsitektur REST API serta penerapan metode Scrum yang adaptif, diharapkan aplikasi ini dapat menyelesaikan permasalahan yang ditemukan. Hasil dari makalah ini diharapkan dapat dijadikan rujukan bagi pengembangan sistem informasi serupa di industri perbankan, serta memberikan wawasan mengenai penerapan teknologi Spring dan metode Scrum dalam pengembangan aplikasi.

1.2 Ruang Lingkup

Pengembangan aplikasi *backend* BSI PINTER dilaksanakan selama kurang lebih tiga bulan dimulai dari bulan Desember tahun 2023 sampai Februari tahun 2024. Aplikasi BSI PINTER dikembangkan dengan metode Scrum dalam tiga fase *sprint*. Proyek ini dikerjakan oleh tiga tim yang terdiri atas *frontend developer*, *backend developer*, dan QA *tester*. Ketiga tim tersebut tercakup ke dalam tim Scrum dengan *product owner* dan *project manager*.

Tim *frontend developer* mengembangkan aplikasi *mobile* Android dan mengintegrasikan API dari *backend* untuk kebutuhan pada setiap fitur. Sedangkan tim *backend developer* mengembangkan aplikasi *backend* dengan menggunakan *framework* Spring, memastikan tersedianya API yang dibutuhkan oleh *client* (frontend developer), memastikan rancangan API diimplementasikan dan dapat diakses oleh *client*. Tim QA *tester* memiliki tugas untuk melakukan pengetesan aplikasi berdasarkan skenario yang telah didefinisikan, memastikan setiap fitur berfungsi sesuai dengan spesifikasi serta melaporkan *bug* atau masalah yang muncul pada hasil pengujian.

Laporan ini mencakup pembahasan mengenai pengembangan aplikasi backend BSI PINTER menggunakan framework *Spring*. Laporan ini berfokus pada penyajian alur pengembangan aplikasi BSI PINTER menggunakan *framework* Spring serta penerapan metode Scrum dalam pengembangannya. Fokus utama laporan ini adalah penyajian implementasi aplikasi secara teknis dengan mengasosiasikan penerapan metode scrum sesuai siklus hidupnya.

Penulis melakukan pengembangan aplikasi dalam tiga fase *sprint*. Sebelum *sprint* dieksekusi penulis berkontribusi dalam penentuan *product backlog* berkolaborasi dengan seluruh tim dan *product owner*. Penulis sebagai *backend developer* berkontribusi dalam seluruh siklus hidup *scrum* (Geoff Watts, 2013). Pada proyek ini, kontribusi penulis mencakup beberapa aspek dalam pengembangan aplikasi *backend* BSI PINTER sebagai berikut:

- a. Memberikan saran dan terlibat diskusi terkait penerapan *Scrum* pada proyek.
- b. Mengembangkan fitur autentikasi dan registrasi pada proyek dengan menggunakan *Spring Security*.
- c. Mengembangkan fitur kuis untuk insan BSI.
- d. Mengembangkan fitur kuis untuk admin insan BSI.
- e. Mengembangkan fitur pengelolaan ruang *meeting* untuk insan BSI.
- f. Melakukan integrasi *Google Cloud* untuk penyimpanan file media.

Terdapat beberapa batasan dalam pengembangan proyek yang perlu diperhatikan. Batasan dalam pengembangan proyek ini adalah bahwa Pengembangan fitur ruang *meeting* pada *sprint* tiga memiliki cakupan untuk sisi *user* saja sesuai dengan *request* *product owner*.

1.3 Tujuan

Tujuan dari pengembangan *back end* aplikasi BSI PINTER adalah untuk menyediakan solusi digital yang terintegrasi untuk memudahkan insan BSI dalam akses informasi produk BSI, serta efisiensi operasional pada pengelolaan ruang *meeting*. Aplikasi ini dirancang untuk meningkatkan efektifitas dan efisiensi operasional karyawan melalui penyajian informasi yang terpusat dan fitur evaluasi yang sistematis. Hasil dari pengembangan aplikasi *backend* akan digunakan oleh *frontend developer* untuk integrasi pada aplikasi *mobile Android*, sehingga semua fitur yang telah dikembangkan di *backend* dapat diakses oleh pengguna akhir melalui antarmuka pengguna yang interaktif dan responsif.

1.4 Manfaat

Hasil dari kegiatan magang ini diharapkan bisa menjadi referensi dan wawasan dalam mengembangkan fitur dan implementasi teknologi serupa. Alur implementasi teknis dan arsitektur pada aplikasi dapat dijadikan acuan untuk mengembangkan aplikasi yang kokoh serta aman dengan menggunakan *framework Spring*. Disisi lain penerapan *Scrum* yang dijelaskan dalam laporan ini diharapkan dapat memberikan wawasan tentang bagaimana metode ini dapat

meningkatkan efisiensi dan efektifitas dalam pengembangan aplikasi. Secara keseluruhan, proses pengembangan dan teknologi yang digunakan yang disampaikan pada laporan ini bisa menjadi sumber informasi dan referensi untuk organisasi dan individu yang ingin menerapkan teknologi dengan metode dan teknologi serupa berdasarkan pengalaman dan hasil implementasi oleh penulis ketika magang.

Secara khusus, aplikasi BSI PITNER memiliki manfaat kepada IT Development Group untuk memberikan kemudahan akses informasi bagi seluruh insan BSI, serta memberikan kemudahan dan transparansi mengenai pengelolaan ruang *meeting*. BSI PINTER memiliki manfaat sebagai berikut:

1. Meningkatkan efisiensi operasional

Aplikasi ini dirancang untuk mempermudah akses informasi produk dan manajemen operasional, termasuk pengelolaan ruang *meeting*.

2. Menyediakan solusi terintegrasi

Sebagai solusi terpadu yang mengintegrasikan seluruh informasi dan kebutuhan operasional dalam satu aplikasi, memudahkan insan BSI dalam mendapatkan informasi yang dibutuhkan secara cepat dan akurat.

3. Mendukung digitalisasi

Mendukung langkah strategis BSI dalam upaya melakukan proses digitalisasi dalam lingkup perbankan, terutama dalam hal akses informasi, evaluasi dan kebutuhan operasional insan BSI.

1.5 Sistematika Penulisan

Sistematika penulisan skripsi ini terdiri dari lima bab yang masing-masing memiliki titik berat yang berbeda, tetapi setiap bab saling memiliki keterkaitan satu sama lain. Sistematika penulisan pada laporan ini adalah sebagai berikut:

- a. BAB I: Pendahuluan

Bab ini menjelaskan mengenai latar belakang, ruang lingkup, tujuan, manfaat dan sistematika penulisan.

- b. BAB II: Landasan Teori dan Tinjauan Pustaka

Bab ini menjelaskan teori-teori yang berkaitan dengan pengembangan proyek yang berhubungan dengan penggunaan *Spring framework* untuk pengembangan aplikasi *backend* serta teori yang berkaitan dengan penggunaan metode Scrum pada pengembangan *software*.

c. BAB III: Pelaksanaan Magang

Bab ini menjelaskan tentang kegiatan dan proses implementasi proyek dengan metode Scrum dalam kegiatan magang. Sebagian besar proses dan hasil dari proyek BSI PINTER disampaikan pada bab ini, khususnya di subbab 3.3

d. BAB IV: Refleksi Pelaksanaan Magang

Bab ini membahas mengenai refleksi dan hasil yang diperoleh selama pelaksanaan magang pada proyek aplikasi *backend* BSI PINTER.

e. BAB V: Penutup

Bab ini membahas mengenai kesimpulan dan saran dari seluruh pembahasan yang telah disajikan pada setiap bab sebelumnya.

BAB II

LANDASAN TEORI

2.1 Backend

Backend, yang dikenal juga sebagai "*server side*," adalah komponen tak terlihat dari aplikasi web yang bertanggung jawab untuk mengelola logika bisnis, penyimpanan data, dan aspek keamanan, sehingga memastikan aplikasi beroperasi dengan baik dan efektif (Ken, 2023). Dalam ranah *backend* terdapat aspek penting yaitu *data layer* atau *database*, ia berperan penting dalam membentuk fondasi infrastruktur pada *software* untuk menyimpan dan mengelola data penting untuk mendukung fungsionalitas aplikasi (Hubli & Jaiswal, 2023). Aplikasi backend menyediakan akses ke *client* (pada umumnya frontend) dengan komunikasi melalui API (Madurapperuma, Shafana, & Sabani, 2022).

2.2 REST API

API (*Application Programming Interface*) adalah sekumpulan aturan dan mekanisme yang memungkinkan berbagai aplikasi perangkat lunak dapat berkomunikasi satu sama lain. API menyediakan cara bagi *developer* untuk mengakses fungsi dan data dari layanan lain tanpa harus memahami detail implementasi internalnya. Dalam konteks web, API umumnya menggunakan protokol HTTP untuk mengirim dan menerima data dalam format seperti JSON dan XML, memungkinkan integrasi antar sistem yang berbeda. API dirancang untuk mempermudah pengembangan aplikasi, mengemat waktu, dan memastikan konsistensi serta keamanan dalam komunikasi antar sistem (Jin, Sahni, & Shevat, 2018).

REST API adalah sebuah API yang menerapkan aturan dan batasan pada arsitektur REST sesuai dengan prinsip-prinsip yang diuraikan oleh Roy Fielding (Fielding, 2000). Arsitektur REST memungkinkan sebuah aplikasi dapat berinteraksi dengan RESTful *web services*. REST pada dasarnya adalah sekumpulan aturan atau batasan, bukanlah sebuah protokol maupun standar. Pengembang API dapat mengimplementasikan REST dalam berbagai metode. Ketika *client* melakukan *request* melalui RESTful API, *client* mengirim atau menerima sebuah representasi data melalui sebuah *endpoint* API. Data dapat dikirimkan dengan berbagai format, tetapi pada umumnya format yang digunakan adalah JSON, HTML, dan XLT. JSON adalah format yang paling sering digunakan, karena JSON memiliki interoperabilitas yang baik dan mudah dipahami (Red Hat, 2020).

Konsep utama dibalik REST adalah bahwa setiap *resource* diidentifikasi dengan URL (Uniform Resource Locator) dan dapat diakses serta dimanipulasi melalui metode HTTP standar seperti GET, POST, PUT, dan DELETE. Pada REST dokumen apapun yang dikirimkan oleh *server* dapat disebut sebagai representasi dari resource. Setiap URL dalam REST mengidentifikasi *resource* pada *server*. Ketika *client* membuat sebuah HTTP *request* ke URL, ia akan mendapatkan representasi dari *resource* yang mendasari *server*. REST Bukanlah sebuah protokol, format fail, maupun framework. Ia adalah sekumpulan aturan atau batasan yang didasarkan dari disertasi Roy Fielding (Fielding, 2000) , atau juga dapat disebut dengan Fielding *constraints*. Dalam konsep REST, *resource* dapat berarti apapun yang dapat disimpan dalam komputer: dokumen elektronik, baris dalam *database*, atau hasil dari komputasi algoritma. Satu satunya batasan harus dipenuhi adalah setiap *resource* harus mempunyai URL. *Server* mengirim representasi yang menggambarkan keadaan *resource*, *client* mengirimkan representasi *resource* yang menggambarkan keadaan yang diinginkan oleh *resource* tersebut. Hal itulah kenapa disebut dengan Representational State Transfer. Untuk dianggap sebagai REST sebuah API harus memenuhi prinsip-prinsip berikut menurut buku *RestFul Web APIs* (Richardson, Amundsen, & Ruby, 2013):

1. **Resource:** Dalam REST, segala sesuatu adalah *resource*(Sumber daya). *resource* diidentifikasi menggunakan URL. Sebagai contoh, untuk mendapatkan data tentang pengguna, maka dimungkinkan untuk menggunakan URL seperti “https://sample-api/users/123”. URL tersebut digunakan untuk mendapatkan representasi *resource* data pengguna dengan id 123.
2. **Representasi:** *Resource* dapat direpresentasikan dalam berbagai format, seperti JSON dan XML. Ketika *client* meminta sebuah *resource* dari *server*, *server* mengirim representasi *resource* tersebut dalam format yang disepakati. Dalam REST *server* tidak mengirimkan keadaan sebenarnya dari *resource* melainkan hanya representasinya saja.
3. **Stateless:** Setiap *request* dari *client* ke *server* harus berisi semua informasi yang diperlukan untuk memahami dan memproses *request* tersebut. *Server* tidak boleh memproses informasi status (*session* adalah contoh status) *client* antar *request*. *Request* harus sepenuhnya terisolasi dan hanya memproses *request* secara independen.
4. **Hypermedia:** REST API dapat menyediakan *link* yang relevan dalam *response* untuk memandu *client* melalui berbagai operasi yang tersedia. Hal ini

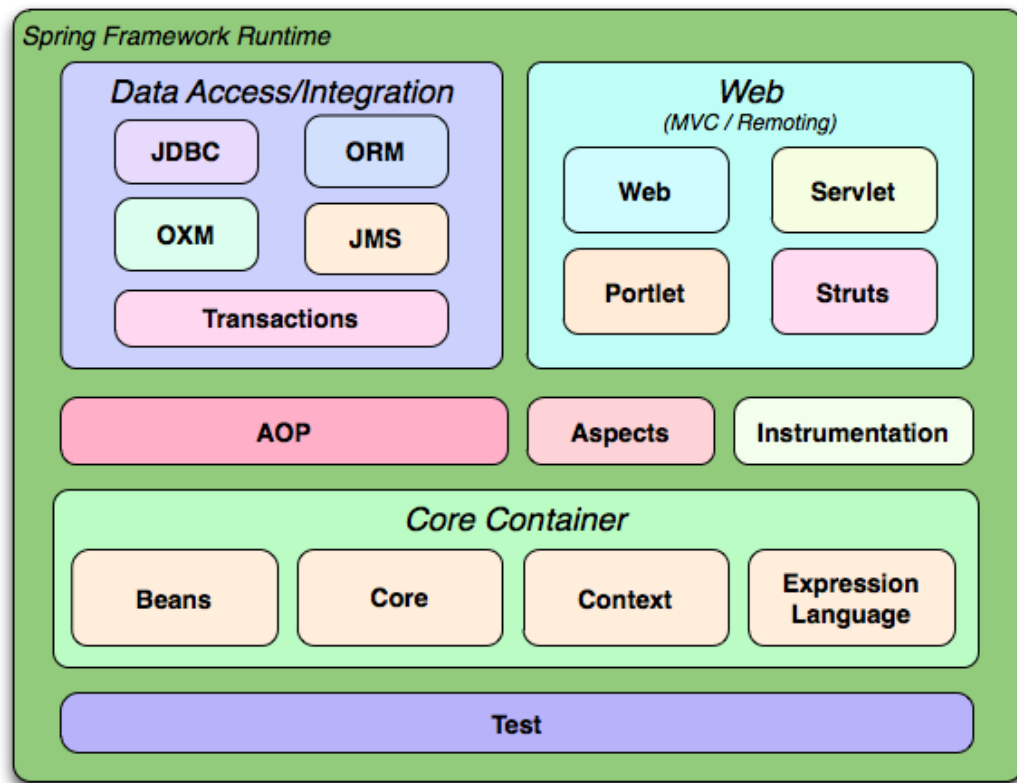
memungkinkan *client* untuk melakukan navigasi aplikasi dengan mengikuti *link* yang diberikan *server*.

5. **URL based interaction:** Setiap interaksi dengan API dilakukan melalui URL yang mewakili *resource* atau sekumpulan *resource*. Struktur dalam URL harus konsisten dan logis.

2.3 Spring Framework

Spring framework merupakan sebuah *framework open source* yang dikembangkan oleh Vmware. Spring menyediakan infrastruktur yang komprehensif dalam mengembangkan aplikasi Java dengan mudah dan cepat. Spring membantu *programmer* dalam mengembangkan aplikasi dengan proses *build* sederhana, *portable*, cepat, dan sistem berbasis JVM yang fleksibel. Fleksibel dalam artian aplikasi Spring dapat dijalankan pada JVM manapun. Spring memungkinkan kita untuk menggunakan modul-modul tertentu saja, sehingga kita tidak perlu menggunakan semua modul Spring dalam aplikasi jika memang tidak diperlukan (Roynardo & Bello, 2017).

Spring framework digunakan untuk membuat aplikasi dengan bahasa pemrograman Java. Spring dibuat sebagai alternatif dari teknologi EJB (Enterprise Java Beans) yang umumnya kurang disukai oleh banyak *programmer*. Manfaat utama penggunaan Spring framework terlepas dari banyaknya modul yang dimiliki, semua modul yang ada pada Spring framework tidak akan dimuat dalam aplikasi jika tidak digunakan. *Developer* dapat memilih modul apa saja yang akan dimuat pada aplikasi, hal ini akan berdampak terhadap signifikansi performa dan jumlah *resource* yang dipakai. Dalam Spring framework, modul Spring Core dan Spring Beans adalah modul terpenting pada aplikasi. Modul tersebut menyediakan fungsionalitas untuk “IOC” (*Inversion of Control*) dan “*Dependency Injection*”. IOC merupakan sebuah paradigma dimana modul tingkat tinggi dapat memilih modul tingkat rendah mana yang mereka gunakan. Hasilnya, Sebagian tanggungjawab akan dialihkan ke framework (Gajewski & Zabierowski, 2019).



Gambar 2.1 Spring framework Runtime

Spring framework terdiri dari banyak fitur yang terorganisasi kedalam 20 modul. Dalam Spring, module Core dan Beans adalah modul fundamental yang mendasari Spring framework termasuk fitur IOC dan *Dependency Injection*. Secara umum Spring framework terdiri atas beberapa modul yang dikelompokkan dalam Core Container, Data Access, Web, AOP (*Aspect Oriented Programming*), dan Test dalam sebuah Spring runtime seperti pada Gambar 2.1. Core Container terdiri atas Beans, Core, Context dan model untuk Expression language. Kelompok tersebut merupakan fondasi utama dari Spring framework termasuk didalamnya fitur IOC (*Inversion Of Control*) dan *dependency injection*. Modul Data Access/Integration terdiri dari dari JDBC, ORM, OXM, JMS dan modul untuk *transactions*. Lapisan web terdiri dari Web, Web-Servlet, dan Web-Portlet Modul. Modul AOP dan instrumentation digunakan untuk melakukan fitur *Aspect Oriented Programming* yang menyediakan integrasi dengan AspectJ. Modul Test menyediakan untuk keperluan pengujian pada aplikasi yang mendukung integrasi dengan Junit. Modul test juga menyediakan mock object yang digunakan untuk melakukan testing terisolasi.(Spring, n.d.)

Dalam buku Pro Spring (Cosmina, Harrop, Schaefer, & Ho, 2017), dijelaskan bahwa Spring merupakan sebuah *framework opensource* yang dirancang untuk memfasilitasi

pengembangan aplikasi berbasis Java yang efisien dan terstruktur. Framework ini mengadopsi prinsip *inversion of control* dan *dependenccy injection* yang memungkinkan aplikasi dapat mengelola objek-objek didalamnya dan mengatur siklus hidupnya. Terdapat Spring Container yang mengelola pembuatan dan penyuntikan dependensi, sehingga mengurangi keterkaitan antar komponen dan meningkatkan modularitas dan performa pengujian. Spring menawarkan modularitas yang tinggi dengan membagi fungsionalitasnya kedalam berbagai modul yaitu Core Container, Aspect Oriented Programming, Data Access/Integration, dan Web. AOP memungkinkan pemisahan logika seperti *logging* atau *transaction* dari logika bisnis utama, membuat kode menjadi lebih bersih dan terorganisir. Selain itu, Spring menyediakan dukungan manajemen transaksi yang menyederhanakan konfigurasi dan pengelolaan transaksi pada aplikasi. Framework ini mendukung integrasi dengan berbagai modul atau *framework* lain seperti JPA, Hibernate, dan JMS. Pada umumnya Spring dikombinasikan dengan Spring Boot untuk memudahkan pengembangan dengan memanfaatkan fitur konfigurasi secara otomatis.

2.4 Spring Boot

Spring Boot adalah sebuah *framework* Java yang memudahkan pengembang aplikasi dalam membuat aplikasi berbasis Spring dengan konfigurasi yang minimal. Spring Boot pada dasarnya merupakan ekstensi konfigurasi untuk platform Java Spring yang dirancang untuk meminimalisir konfigurasi dalam pembuatan aplikasi berbasis Spring. Dengan menggunakan Spring Boot pengembang dapat membuat sebuah aplikasi yang *standalone* atau mandiri dengan usaha yang minimal tanpa melakukan konfigurasi yang kompleks (Spring, n.d.).

Spring Boot merupakan kerangka kerja dalam ekosistem Spring yang dirancang untuk menyederhanakan pengembangan aplikasi berbasis Spring dengan menghilangkan kebutuhan konfigurasi yang kompleks dan menawarkan pendekatan otomatis yang efisien. Secara otomatis Spring Boot melakukan konfigurasi berdasarkan dependensi yang ada pada *classpath*, sehingga mempermudah pengaturan dan pengembangan aplikasi. Fitur utama dari Spring Boot adalah *auto-configuration*. Fitur tersebut dapat menkonfigurasi komponen secara otomatis. Spring Boot menyediakan fitur *starters* yang berupa paket dependensi, sehingga tidak perlu menambahkan dependensi satu per satu. Spring Boot memiliki *embeded server* yang terintegrasi secara otomatis ketika kita menggunakannya, memungkinkan aplikasi dijalankan secara mandiri tanpa memerlukan konfigurasi *server* secara terpisah (Walls, 2019).

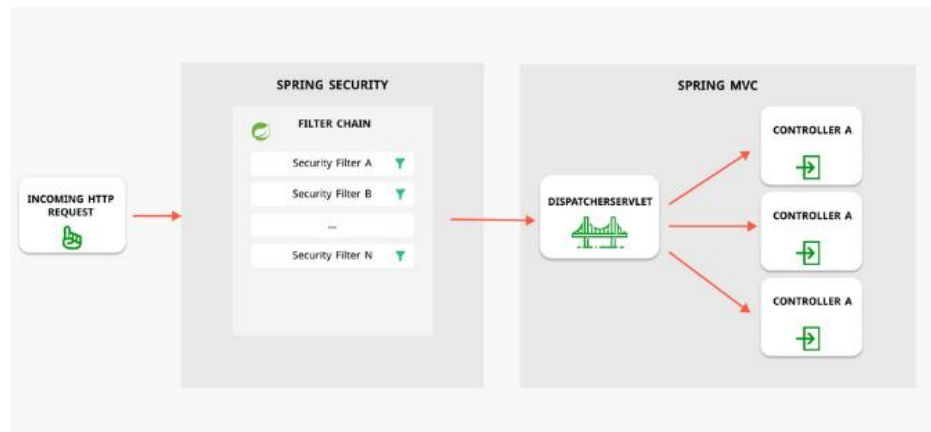
Salah satu fitur Spring Boot yang membantu pengembangan aplikasi Spring adalah kemampuannya untuk menyederhanakan proses pengaturan dan penyebaran aplikasi melalui

konsep *auto-configuration* dan *starter dependencies*. Dalam bukunya *Spring Boot in Action* (Craig, 2016), Walls menjelaskan bahwa Spring Boot menyediakan berbagai starter, yang mencakup konfigurasi umum seperti Spring Data dan Spring MVC, sehingga pengembang tidak perlu melakukan konfigurasi berulang secara manual. Selain itu, Spring Boot mendukung sejumlah *embedded server*, seperti Tomcat, Jetty, dan Undertow yang secara otomatis terintegrasi. Ini memungkinkan aplikasi berjalan tanpa perlu mengonfigurasi *server* dari luar. Untuk membuat aplikasi Spring yang *scalable* dan siap produksi dengan lebih cepat, kemampuan ini sangat membantu dalam mengurangi kerumitan dan meningkatkan produktivitas pengembang.

2.5 Spring Security

Spring security merupakan sebuah *framework* yang kokoh untuk mengamankan aplikasi berbasis Spring. Konsep inti dari Spring Security adalah “*Security Filter Chain*” yang terdiri dari serangkaian filter yang diproses oleh aplikasi untuk setiap *request* HTTP. Filter-filter ini berfungsi untuk melakukan autentikasi, otorisasi, dan perlindungan terhadap serangan keamanan lain seperti CSRF dan XSS. Spring Security memungkinkan konfigurasi yang fleksibel baik melalui XML atau kode Java, serta menyediakan integrasi yang mulus dengan teknologi autentikasi eksternal seperti LDAP, OAuth, dan CAS. Dengan fitur-fitur ini, Spring Security memberikan lapisan keamanan yang kokoh untuk aplikasi Spring, memungkinkan pengembang untuk pada pengembangan logika bisnis tanpa khawatir terhadap masalah keamanan (Spilca, 2020).

Dalam Spring Security, Security Filter Chain adalah mekanisme utama yang menjaga keamanan aplikasi dengan menjalankan sejumlah filter untuk setiap *request* HTTP. Setiap filter dalam rantai melakukan fungsi tertentu. Fungsi tersebut termasuk dalam melakukan autentikasi, proteksi terhadap CSRF, dan pemrosesan *login* serta *logout*. FilterChain dieksekusi secara berurutan, memungkinkan pengembang dapat mengontrol aliran *request* dengan menambahkan, menghapus, atau mengubahnya sesuai dengan kebutuhan. Urutan filter yang tepat dalam *filterchain* sangat penting karena akan berdampak pada keamanan dan perilaku aplikasi dalam memproses *request* HTTP. Oleh karena itu, konfigurasi filter dalam Filter Chain merupakan aspek penting yang harus diperhatikan (Knutson, Winch, & Mularien, 2017).



Gambar 2.2 Spring Security Filter Chain

Gambar 2.2 merupakan diagram dari Security Filter Chain, menunjukkan bagaimana alur Spring Security bekerja. FilterChain merupakan konsep inti bagaimana Spring Security bekerja. Pertama, *client* mengirimkan request HTTP yang kemudian disaring oleh Spring security melalui FilterChain. Request yang lolos melalui proses penyaringan akan diteruskan ke *controller* yang sesuai melalui Dispatcher Servlet. Filter tersebut dapat digunakan untuk melakukan fungsi tertentu seperti melakukan autentikasi dan otorisasi.

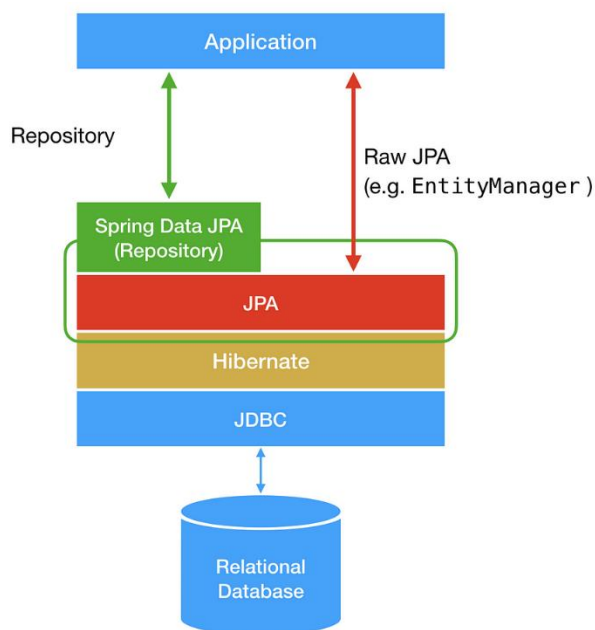
Penggunaan Spring Security tidak lantas menjadikan aplikasi menjadi aman sepenuhnya dari segala acaman keamanan. Menurut Islam (Islam et al., 2020) Proses autokonfigurasi yang dilakukan oleh Spring Boot pada Spring Security masih memunculkan celah keamanan jika tidak ditangani dengan sesuai. Untuk memaksimalkan kemampuan Spring Security, Islam juga menekankan diperlukan proses konfigurasi lebih lanjut seperti memberikan rekomendasi pengaturan masa berlaku token yang lebih pendek, penggunaan algoritma *hashing* yang lebih kuat, dan menerapkan kebijakan pembatasan permintaan. Pengembang dituntut untuk adaptif terhadap kemungkinan celah keamanan baru dan menyesuaikan dengan setiap versi pada Spring Security.

2.6 Spring Data JPA

Spring data JPA merupakan salah satu modul dari Spring Framework untuk mengimplementasikan lapisan akses data dengan menggunakan teknologi JPA (Java Persistence API). Dalam buku *Java Persistence with Hibernate* (Bauer, King, & Gregory, 2016), dijelaskan bahwa Spring Data JPA menyediakan abstraksi tingkat tinggi diatas JPA sehingga mengurangi *boilerplate code* yang diperlukan untuk pengembangan. Modul ini memiliki fitur utama termasuk *repository* berbasis *interface* yang memungkinkan *developer* untuk mendefinisikan kueri hanya dengan mendeklarasikannya sebagai method abstrak,

kemudian implementasinya dihasilkan secara otomatis oleh Spring pada *runtime*. Dengan modul ini, *developer* dapat lebih fokus kepada logika bisnis aplikasi tanpa memikirkan detail implementasi untuk akses data. Modul ini mendukung fitur JPA seperti pembuatan entitas, relasi, dan JPQL (*Java Persistence Query Language*), serta integrasi dengan teknologi yang mengimplementasikannya seperti Hibernate. Hibernate memperluas fungsionalitas dan performa dalam pengelolaan *database*.

Spring Data JPA merupakan sebuah framework yang bertanggungjawab untuk melakukan *object-relational mapping*, memetakan objek atau model Java kedalam tabel spesifik pada *database* (Bonteanu & Tudose, 2024). Spring Data JPA menyederhanakan kebutuhan data akses menggunakan Java Persistence API (JPA). Ia merupakan abstraksi yang bekerja diatas ORM seperti Hibernate. Ia pada dasarnya merupakan spesifikasi untuk ORM di Java yang menyediakan cara standar untuk berinteraksi dengan *database* (Fisher & Murphy, 2016).



Gambar 2.3 Diagram Lapisan Aplikasi Spring Data JPA

Penggunaan Spring Data JPA sangat berkaitan erat dengan ORM. JPA memerlukan *framework* yang mengimplementasikan spesifikasi standarnya. Pada umumnya *framework* hibernate merupakan pilihan yang paling populer sebagai implementasi dari JPA. Diagram Gambar 2.3 menunjukkan keterkaitan Spring Data JPA dan Hibernate. Hibernate merupakan implementasi dari JPA yang berfungsi sebagai ORM yang menerjemahkan kelas Java menjadi tabel *database*, menangani pembuatan keuri dan eksekusi SQL. Hibernate memastikan keberlangsungan objek yang disimpan dan memastikan sinkronasi *database* pada aplikasi.

Pada layer paling atas (Application) merupakan tempat dimana logika bisnis aplikasi berada, berinteraksi dengan *email* melalui Repository Spring Data JPA. Spring data JPA memanfaatkan JPA yang menyediakan spesifikasi standar bagaimana objek Java berinteraksi dengan *relational database*. Hibernate mengimplementasi spesifikasi dari JPA untuk berinteraksi dengan *database* melalui JDBC. Hibernate menggunakan JDBC untuk mengeksekusi pernyataan SQL untuk melakukan operasi CRUD pada *database*.

2.7 Scrum

Scrum adalah kerangka kerja (*framework*) untuk mengelola kerja yang kompleks, sering digunakan dalam pengembangan perangkat lunak, tetapi juga bisa diterapkan dalam berbagai jenis proyek. Scrum didesain untuk membantu tim bekerja secara kolaboratif dalam lingkungan yang kompleks, berubah, dan penuh ketidakpastian (Sutherland, 2014). Berdasarkan Scrum Guide (Schwaber & Sutherland, 2020) Scrum bukan merupakan metodologi yang kaku, melainkan *framework* ringan yang memberikan panduan mengenai peran, *event*, dan artefak yang diperlukan untuk mengelola proyek. Scrum mendorong tim untuk bekerja secara iteratif dan bertahap. Scrum memiliki beberapa nilai dan prinsip utama sebagai berikut:

1. **Transparent:** Semua aspek pekerjaan harus jelas dan dapat dilihat oleh seluruh anggota tim.
2. **Inspection:** Pemeriksaan membantu tim scrum untuk mendeteksi varian, masalah, dan penyimpangan dalam proses pengembangan proyek yang sedang dijalankan.
3. **Adaptation:** Berdasarkan hasil inspeksi, tim harus siap beradaptasi untuk meningkatkan proses atau produk. Tim diharapkan beradaptasi dan mempelajari hal baru berdasarkan hasil pemeriksaan.

Hal yang fundamental dalam Scrum adalah tim yang terdiri dari lingkup kecil yang terdiri atas *Scrum Master*, *Product Owner* dan *Developer*. Mereka adalah satu kesatuan yang terdiri atas profesional yang berfokus pada satu tujuan yaitu *Product Goal*. Masing-masing peran memiliki tanggungjawab sebagai berikut:

4. **Product owner:** bertanggungjawab terhadap manajemen *product backlog* dan memastikan bahwa produk yang dikembangkan sesuai dengan tujuan dan kebutuhan produk.
5. **Scrum master:** Bertugas untuk memastikan pemahaman dan implementasi Scrum oleh tim, serta membantu menangani hambatan dan kendala yang dialami oleh tim.

6. **Developer:** Bertanggungjawab untuk mengembangkan produk dan menghasilkan *increment* pada setiap Sprint.

Dalam scrum terdapat istilah artefak yang merujuk pada produk, dokumen atau hasil kerja selama proses pengembangan sebagai transparansi dari pekerjaan yang sedang berlangsung. Artefak utama dalam Scrum meliputi:

1. **Product Backlog:** daftar prioritas semua fitur, perubahan dan perbaikan yang diperlukan oleh produk. *Product backlog* disusun dan dikelola oleh *Product Owner*.
2. **Sprint Backlog:** bagian dari *Product Backlog* yang dipilih untuk dikerjakan selama satu Sprint.
3. **Increment:** Merupakan hasil kerja yang terintegrasi, merupakan hasil dari eksekusi Sprint yang memenuhi *acceptance criteria* dan menjadi nilai tambah bagi produk.

Sprint dalam Scrum merupakan wadah bagi seluruh *event*. *Event* pada Scrum merupakan sebuah kesempatan formal untuk memeriksa dan menyesuaikan terhadap artefak scrum. *Event* tersebut diperlukan untuk memungkinkan adanya transparansi dalam proses pengembangan. Setiap *event* memiliki tujuan khusus dan dirancang untuk meningkatkan kolaborasi dan efisiensi tim Scrum. *Event* memberikan struktur pada Sprint dan memastikan bahwa tim tetap berada di jalur yang benar dan dapat memperbaiki proses secara terus menerus. Hal tersebut juga memastikan bahwa pelajaran yang diperoleh dari pengalaman sebelumnya dapat diterapkan pada pekerjaan mendatang. Dalam Scrum, event terbagi lima bagian seperti berikut:

1. **Sprint:** Merupakan inti dari Scrum dimana ide dikonversi menjadi luaran yang bernilai. Sprint merupakan periode waktu tertentu (satu sampai empat minggu) dimana seluruh tim bekerja untuk menyelesaikan *task* yang diberikan.
2. **Sprint Planning:** Merupakan *event* yang dilaksanakan sebelum fase sprint dimulai pada setiap sprint. Tujuan utama dari *event* ini adalah untuk menentukan Sprint Goals, memilih *Product Backlog items* untuk Sprint, dan membuat rencana untuk pengembangan aplikasi sesuai *Product Backlog* yang dipilih.
3. **Daily Scrum:** Pada *event* ini tim melakukan rapat singkat (biasanya dalam waktu 15 menit) untuk memeriksa kemajuan dari tim terhadap Sprint Goals dan menyesuaikan dengan *Product Backlog* jika diperlukan. Pada tahap ini, tim meninjau kemajuan dari setiap anggota berdasarkan *task* yang diselesaikan, *task* yang sedang berjalan, dan rencana yang akan dilakukan pada hari selanjutnya. *Event* ini juga ditujukan untuk mengidentifikasi munculnya kendala atau *blocker* yang menghambat progress tim.

4. **Sprint Review:** Merupakan salah satu *event* yang penting dalam scrum dimana tim Scrum dan *stakeholder* memeriksa luaran dari Sprint yang telah selesai dan beradaptasi terhadap hasil *review* pada event ini. Pada *event* ini, tim memeriksa kemajuan Sprint terhadap *acceptance criteria* yang sudah didefinisikan pada Product Backlog. Tim juga mendiskusikan kendala yang dihadapi dan bagaimana mereka menangani dan menyelesaikan kendala tersebut. Product backlog juga dapat disesuaikan jika dibutuhkan.
5. **Sprint Retrospective:** Event ini dilaksanakan setelah Sprint Review dan sebelum Sprint Planning. Tujuan dari event ini adalah untuk melakukan refleksi terhadap apa yang sudah berjalan dengan baik, mengidentifikasi kendala dan menemukan area dimana tim dapat melakukan peningkatan. Pada event ini, tim juga mendiskusikan mengenai aspek negatif dan aspek positif pada penerapan Scrum, menyediakan ruang yang aman bagi tim untuk berkomunikasi secara terbuka dan meningkatkan proses kolaborasi. Goal dari *event* ini, tim mempunyai rencana tindak lanjut untuk meningkatkan efektifitas dari tim dan membuat penerapan Sprint selanjutnya menjadi lebih efektif. Dengan melakukan perbaikan secara berkelanjutan, tim dapat meningkatkan performa dan lingkungan kerja secara bertahap.

BAB III

PELAKSANAAN MAGANG

3.1 Aktivitas Magang

Penulis menjalani proses magang pada program BSI Digilab Internship Programme. Program tersebut merupakan program kerjasama magang kampus merdeka B2B antara PT Bank Syariah Indonesia dengan Universitas melalui unit kerja IT Development Group (IDG). Program ini diselenggarakan dalam rangka percepatan penyelesaian proyek dan pengembangan produk IT dan digital yang dilaksanakan pada Digital Lab BSI yang berada di Yogyakarta.

Pada program magang tersebut, penulis berkesempatan untuk terlibat sebagai *Backend Developer* sebagai peserta magang IDG. Pada program magang ini, penulis menjalani beberapa tahapan sesuai ketentuan program seperti berikut:

1. **Induction:** Merupakan tahap pengenalan lingkungan magang, mentor dan *briefing* program yang berlangsung selama satu hari, dilaksanakan secara *offline class* di kantor Digilab BSI Yogyakarta.
2. **General Banking:** Tahap materi dasar perbankan secara umum dan perbankan syariah secara khusus. Tahap ini bertujuan untuk memberikan pemahaman yang komprehensif secara teori atas model bisnis bank syariah.
3. **OJT Training:** Peserta magang melakukan *On the Job Training* di kantor cabang/kantor cabang pembantu di jaringan area Yogyakarta. Tahap ini bertujuan untuk memberikan pengalaman secara langsung di lapangan untuk memberikan pemahaman yang komprehensif atas proses bisnis bank Syariah dan menggali *IT Issue Takeways*.
4. **Specialist IT:** Peserta magang mengikuti inclass training secara spesifik mengenai materi IT oleh *principal* atau konsultan IT di BSI. Peserta dijelaskan mengenai modul IT secara umum dan teknologi yang diimplementasikan dalam perbankan khususnya BSI.
5. **OJT:** Peserta membangun pengalaman dan terlibat langsung dalam implementasi teknologi IT di perbankan syariah sesuai dengan bidang pilihan secara *offline* selama kurang lebih 18 pekan.

Pada program magang ini, penulis dibimbing oleh beberapa mentor dari IDG dan diawasi oleh satu koordinator lapangan yaitu Bapak Taufik Hidayat yang bertugas untuk mengawasi

jalannya proses magang. Sedangkan mentor yang bertugas selama pengembangan proyek adalah sebagai berikut:

1. Bapak Andrika Zainal Ibrahim, UI/UX designer IDG: memberikan arahan mengenai proses perancangan UI/UX dan memberikan arahan mengenai alur kerja aplikasi.
2. Bapak Eki Iskandar Yusandhi, Innovator & Android Developer IDG: Memberikan arahan pada proses pengembangan Aplikasi Android. Beliau juga mendampingi anggota tim magang dalam memahami proses scrum yang dilaksanakan.
3. Bapak Muhammad Faiq Ardyanto Putro, Innovator & Website Developer IT Application Support: memberikan arahan kepada tim magang QA dalam membuat skenario pengujian pada aplikasi *frontend* maupun *backend*.
4. Bapak Ramaditya Dimas Razandi, Innovator & Backend Developer IDG: Mendampingi proses pengembangan proyek dari awal sampai akhir. Memberikan pendampingan kepada anggota tim magang backend dalam setiap siklus pengembangan seperti fase perencanaan, desain, implementasi dan pengujian. Beliau juga memberikan arahan mengenai proses integrasi antara backend dan frontend, bagaimana backend dan frontend saling bekerjasama dalam mengembangkan sebuah fitur.

Penulis mendapatkan beberapa arahan mengenai proses pengembangan *software* selama proses magang. Pada magang tersebut, penulis dilibatkan dalam tim *Backend Developer* untuk mengembangkan aplikasi internal perusahaan yaitu BSI PINTER (Sistem Pusat Informasi Terpadu). Proses pengembangan aplikasi tersebut dilaksanakan selama kurang lebih 4 bulan. Aplikasi BSI PINTER dikembangkan dengan menggunakan metodologi Agile dengan metode Scrum.

3.2 Manajemen Proyek

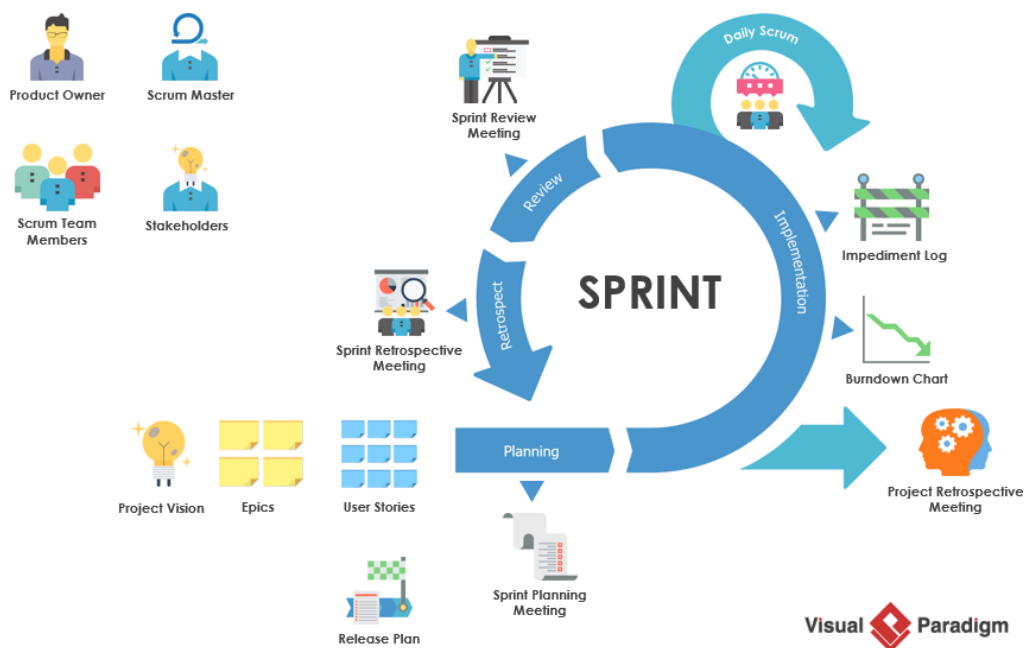
3.2.1 Penerapan Metode Scrum

Aplikasi BSI PINTER dikembangkan dengan menggunakan metodologi Agile dengan metode Scrum. Pelaksanaan Scrum pada BSI PINTER didasarkan pada panduan Scrum Guide (Schwaber & Sutherland, 2020) dengan menerapkan seluruh *lifecycle* pada Scrum mulai dari pengumpulan *product backlog* sampai dengan *event* terakhir yaitu Sprint Retrospective. Penggunaan metode agile, khususnya metode Scrum dipilih dengan pertimbangan yang matang, metode Scrum memberikan fleksibilitas dan transparansi yang diperlukan dalam

menangani perubahan kebutuhan dan kompleksitas proyek. Dalam konteks ini, tahap awal pengembangan aplikasi BSI PINTER memungkinkan pengumpulan *product requirements* yang dinamis sesuai dengan keputusan Product Owner.

Dalam Scrum terdapat beberapa tim yang berperan penting dalam menjalankan proses pengembangan dan memastikan tujuan dari proyek tercapai dengan efektif. Pada proyek ini, tim Scrum terdiri dari *Product Owner*, *Product Manager*, dan tim *Development*. Tim *Development* terdiri atas developer yaitu *Frontend Developer* dan *Backend Developer* serta tim *Quality Assurance*. Product manager memiliki tanggungjawab untuk menentukan strategi dan *roadmap* proyek, bekerjasama dengan *project manager* untuk mengarahkan pengembangan proyek. Product owner bertanggungjawab dalam membuat, memelihara, dan mengatur daftar Product Backlog. Tim developer bertugas untuk mengembangkan aplikasi, merealisasikan product backlog kedalam bentuk fitur yang sesuai. Pada konteks Scrum hasil fitur yang dikembangkan merupakan sebuah increment yaitu kumpulan dari semua Sprint Backlog yang telah selesai. Pada proyek ini tim developer terdiri dari dua yaitu frontend developer yang memiliki jobdesk untuk mengembangkan aplikasi mobile android dan backend developer yang mengembangkan aplikasi *backend* dengan menggunakan Spring. Kemudian hasil dari pengembangan fitur dilakukan pengujian oleh tim Quality Assurance. Hasil pengujian menentukan kelayakan aplikasi untuk fase *production*. Seluruh tim bersinergi dalam mewujudkan *goals* dalam setiap Sprint-nya.

The Agile – Scrum Framework



Gambar 3.1 Scrum Lifecycle

sumber: <https://www.visual-paradigm.com/servlet/editor-content/scrum/how-scrum-team-works/sites/7/2018/11/the-agile-scrum-framework-1.png>

Fase pengembangan aplikasi dengan menggunakan Scrum terdiri atas beberapa tahapan sesuai dengan *lifecycle* pada Gambar 3.1. Pada fase awal, *product manager* menentukan visi product berdasarkan masukan dan kebutuhan dari *stakeholder*. Selanjutnya *product owner* menentukan daftar *product backlog* berdasarkan kebutuhan dari *stakeholder*. *Product backlog* disini adalah daftar *user stories* yang menggambarkan fitur dan kebutuhan pengguna. Setiap *user story* memiliki *acceptance criteria* yang jelas untuk memastikan kebutuhan dari *stakeholder* sesuai dan dan terpenuhi. Secara berkala, *product owner* mengelola dan memperbarui *product backlog* agar tetap relevan dengan visi yang telah ditetapkan.

Pada fase awal Scrum, terdapat daftar Epics yang disusun oleh *product owner*. Epics merupakan *backlog item* skala besar yang merupakan kumpulan dari beberapa *user stories* yang saling terkait yang membentuk satu kesatuan gambaran fitur aplikasi. Daftar *user stories* pada Epics kemudian akan diprioritaskan untuk masuk kedalam Sprint Backlog untuk dikerjakan pada fase Sprint.

Sprint Planning Meeting

Pada proyek ini terdapat empat Scrum Event yang diterapkan yaitu *Sprint Planning Meeting*, *Daily Scrum*, *Sprint Review Meeting*, dan *Sprint Retrospective*. Keempat *event* tersebut masuk kedalam rangkain Sprint yang dijalankan pada setiap Sprint. Tahap awal pelaksanaan Sprint dimulai dengan *planning event* yaitu *Sprint Planning Meeting*. Pada tahap tersebut, tim merencanakan pekerjaan yang akan dilakukan pada fase Sprint. Dalam sesi tersebut tim memilih beberapa *product backlog* yang nantinya akan dieksekusi dalam Sprint. Tim juga menentukan *Sprint Goals* yang akan menjadi tujuan utama dalam Sprint tersebut. Tim memecah beberapa *product backlog* menjadi beberapa task yang lebih kecil sehingga mudah dikelola. Ini membantu tim untuk melacak kemajuan sampai pada lingkup yang lebih kecil dan detail. Pada tim *backend*, tim memecah *user stories* menjadi beberapa task kecil yang sesuai dengan kebutuhan teknis untuk merealisasikan fitur seperti task untuk pembuatan API Contract, ERD, *endpoint* dan sebagainya. Pada proyek ini, *user stories* di pecah menjadi task yang sedetail mungkin untuk meningkatkan transparansi dan visibilitas antar anggota tim.

Pada fase tersebut tim juga melakukan estimasi waktu dan kapasitas tim dalam memperkirakan berapa banyak *task* yang dapat dikerjakan dalam satu Sprint. Perhitungan estimasi didasarkan pada *Story Points* yang dihitung berdasarkan tingkat kompleksitas pada

user story. Pada akhirnya tim harus menetapkan sebuah komitmen untuk menyepakati *product backlog* yang telah dipilih dan berkomitmen untuk menyelesaikannya.

Eksekusi Sprint

Fase selanjutnya adalah eksekusi Sprint, kurang lebih berjalan selama dua sampai tiga minggu. Tim mengerjakan *task-task* yang sudah di pecah pada fase Sprint Planning sesuai pada Sprint Backlog. Berdasarkan backlog yang telah dipilih, tim *backend* kemudian mendefinisikan desain API contract dan membuat design *email* dalam bentuk ERD (Entity Relationship Diagram). Hasil desain API contract kemudian didiskusikan dan disepakati bersama antara tim frontend developer dan *backend* developer. Setelah API contract disepakati maka *backend* developer kemudian mulai mengembangkan fitur sesuai spesifikasi yang telah ditentukan. Disisi lain, tim QA membuat test script untuk melakukan pengujian terhadap fitur-fitur yang telah dikembangkan. Test script dibuat berdasarkan skenario alur aplikasi dan kebutuhan pada backlog termasuk untuk melakukan validasi sesuai pada acceptance criteria.

Selama fase Sprint tim melakukan *Daily Scrum Meeting* yang dilaksanakan setiap hari. Pelaksanaan event tersebut berjalan kurang lebih dalam waktu 15 menit, disesuaikan dengan waktu Sprint. *Daily Scrum* merupakan manifestasi dari pilar *Inspection*, melacak kemajuan pada Sprint setiap hari. Dalam *event* tersebut setiap anggota tim memberikan *update* mengenai *task* yang dikerjakan, memastikan progres dari setiap tim dapat terlacak. Aktivitas utama dalam *event* tersebut adalah berdiskusi mengenai progress pekerjaan pada setiap anggota tim. Setiap anggota tim menjawab pertanyaan utama yaitu “apa yang telah saya kerjakan kemarin”, “apa yang telah saya kerjakan hari ini”, dan “apakah ada hambatan atau masalah yang menghambat pekerjaan saya”. Hambatan yang teridentifikasi kemudian masuk kedalam *impediment log* untuk kemudian ditangani diluar *event* ini. Dengan terselenggaranya *event* ini, pilar *transparency* dapat tercapai, terdapat transparansi progress dari setiap anggota tim sehingga mengurangi risiko kemungkinan hambatan atau *blocker* yang tidak terdeteksi.

Sprint review meeting

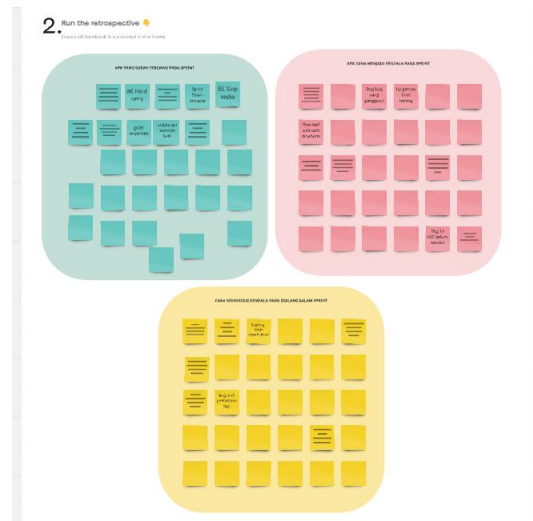
Setelah fase Sprint selesai, tim melaksanakan *Sprint Review Meeting* untuk mengevaluasi hasil kerja yang telah dilakukan selama Sprint. Pada *event* ini tim berkesempatan untuk mendemonstrasikan *increment*, yaitu hasil dari *task* yang telah dikerjakan dan siap untuk dipresentasikan kepada *stakeholders*. Tim menunjukkan *task* apa saja yang berhasil diselesaikan dan memberikan gambaran fungsionalitas yang telah ditambahkan. Dalam

praktiknya, tim *backend* developer menunjukkan hasil *user stories* yang sudah diselesaikan dengan melakukan demonstrasi fitur yang telah dikerjakan, mendemonstrasikan hasil *endpoint* yang telah dikerjakan. Tim *backend* menunjukkan pada setiap *endpoint* bagaimana request dikirimkan dan skenario response yang didapatkan oleh client. Disisi lain, hasil integrasi juga didemonstrasikan oleh tim frontend, menampilkan beberapa halaman yang sudah terintegrasi dengan *backend* dan menunjukkan alur fitur yang telah dikembangkan secara keseluruhan.

Pada *event* ini, *stakeholders* memberikan *feedback* dan saran untuk hasil Sprint yang dapat menjadi bahan penyesuaian pada *backlog* untuk Sprint selanjutnya. *Product owner* juga memvalidasi bahwa *increment* dari tim telah memenuhi *acceptance criteria* yang telah ditentukan pada setiap *user story* atau *product backlog* yang dikerjakan selama fase Sprint. Hasil dari validasi menentukan keputusan *product owner* untuk menyatakan valid atau tidaknya hasil kerja. Apabila valid, maka fitur siap untuk diintegrasikan, sebaliknya jika tidak valid maka *item stories* atau *product backlog* dapat ditambahkan kembali pada Sprint berikutnya untuk perbaikan.

Sprint Retrospective Meeting

Selanjutnya *event* terakhir dari Scrum adalah *Sprint Retrospective Meeting*. Pada *event* tersebut seluruh anggota tim berkumpul untuk melakukan refleksi terhadap proses yang telah dilakukan pada fase Sprint. Event ini bertujuan untuk mengidentifikasi hal-hal yang sudah berjalan dengan baik, hal-hal yang masih perlu diperbaiki, dan potensi perbaikan yang dapat diterapkan pada Sprint selanjutnya. Pada *event* ini tim mengevaluasi aspek non teknis seperti kolaborasi antar tim atau anggota, komunikasi, metode pengerjaan, serta penggunaan *tools* pada Sprint.



Gambar 3.2 Bagian *feedback* Sprint Retrospective Meeting

Pada *Sprint Retrospective Meeting* setiap anggota tim memberikan *feedback* dan masukan mengenai aspek apa saja yang dapat ditingkatkan untuk kinerja tim secara keseluruhan. Sesuai praktik pada magang, penulis melakukan diskusi terbuka untuk mengidentifikasi praktik terbaik yang perlu dipertahankan dan strategi baru yang dapat diimplementasikan untuk meningkatkan efisiensi dan efektifitas pada Sprint selanjutnya. Fokus utama dalam *event* ini adalah untuk melakukan perbaikan secara berkelanjutan terhadap cara kerja tim untuk mencapai hasil yang lebih optimal. Pada saat *event* berlangsung, setiap anggota tim menggunakan sticky notes seperti pada untuk memberikan *feedback* pada bagian apa yang sudah berjalan dengan baik, hal-hal yang masih perlu diperbaiki, dan potensi perbaikan pada sprint selanjutnya. Waktu pengisian *feedback* dilakukan selama kurang lebih lima sampai sepuluh menit. Setelah itu setiap anggota tim menyampaikan apa yang telah dicantumkan pada setiap bagian retrospective, setiap anggota dapat memberikan *feedback* langsung kepada anggota tim lainnya.

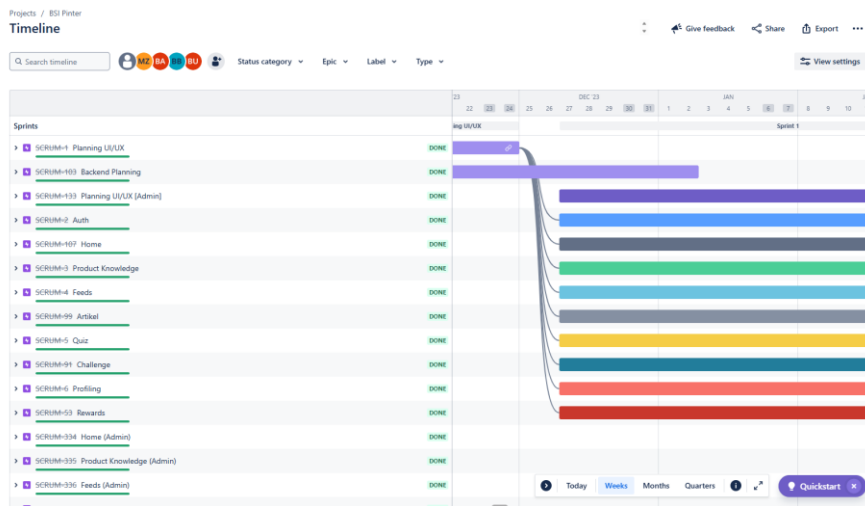
Hasil dari *event* ini adalah daftar aksi konkrit yang akan diambil oleh tim selanjutnya. *Event* ini menekankan bahwa Scrum tidak hanya berfokus pada pengembangan secara teknis saja, tetapi aspek lain untuk pengembangan tim juga diperhatikan. Sesuai pilar *Adaptation*, tim Scrum berkomitmen untuk menyesuaikan dan mengadaptasi proses yang telah dilaksanakan berdasarkan *feedback* dan evaluasi pada *event* ini.

3.2.2 Penggunaan Platform Atlassian

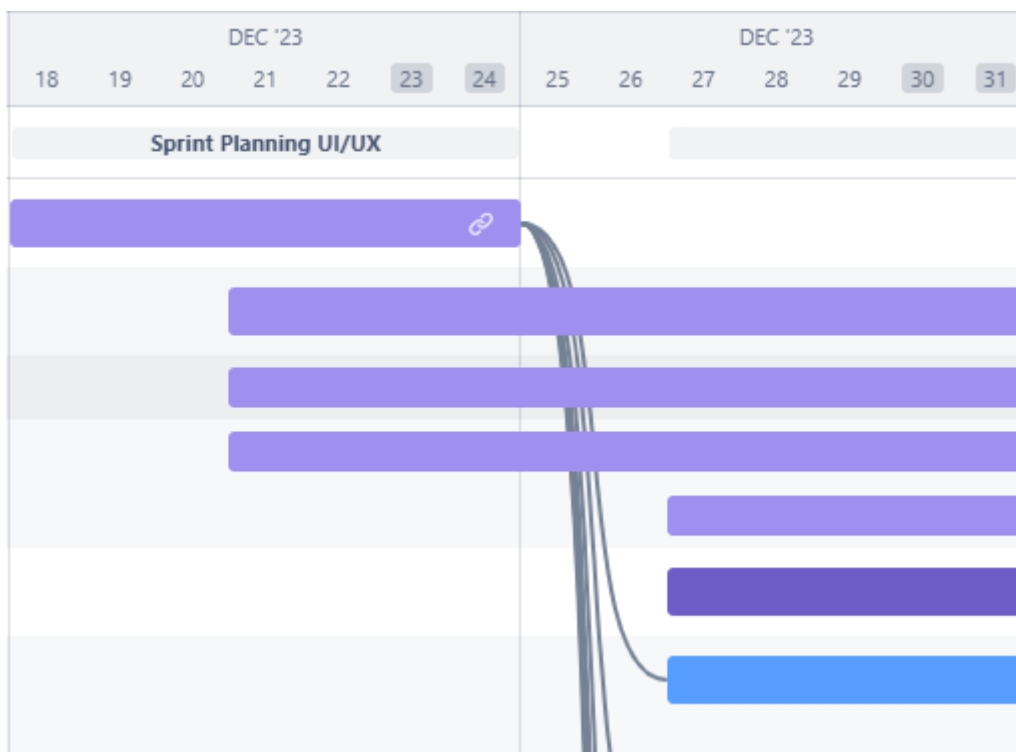
Pada pengembangan aplikasi BSI PINTER tim menggunakan *platform* Atlassian sebagai *tools* untuk manajemen proyek. Aplikasi dari *platform* Atlassian yang digunakan adalah Jira dan Confluence. Jira digunakan untuk memfasilitasi *project management* dengan menggunakan metode Scrum. Aplikasi Jira memfasilitasi implementasi metode Scrum dengan menyediakan beberapa fitur yang mendukung seperti *planning*, *tracking*, *collaboration*, dan *reporting*. Tim juga menggunakan aplikasi Confluence sebagai *tools* untuk melakukan dokumentasi proyek. Confluence digunakan untuk mendokumentasikan seluruh informasi terkait proyek, termasuk spesifikasi produk, dokumen teknis, catatan rapat, dan hasil evaluasi Sprint. Confluence memudahkan dalam pengorganisasian dokumen, memberikan akses terintegrasi pada proyek sehingga memudahkan tim dalam berkolaborasi, berbagi informasi dan mengakses dokumen secara efisien.

Penggunaan Aplikasi Jira

Dalam konteks implementasi Scrum pada sisi pengembangan, aplikasi Jira digunakan dalam *product* dan *project management* pada aplikasi BSI PINTER. Jira didesain secara khusus untuk memfasilitasi manajemen proyek khususnya dalam menerapkan metode Scrum. Untuk perencanaan, fitur *timeline* digunakan dalam memvisualisasikan jadwal proyek dan menentukan prioritas pekerjaan yang harus diselesaikan dalam kurun waktu yang telah ditentukan. Seperti pada Gambar 3.3, *product manager* telah menentukan *timeline* pengembangan yang dikelompokkan dalam Epics. Visualisasi *project* pada *timeline* memungkinkan seluruh anggota tim untuk mendapatkan gambaran pengembangan proyek secara keseluruhan. Dalam pengembangan proyek terdapat sebuah kasus yang memungkinkan sebuah *task* dapat memiliki dependensi pada *task* yang lain. Sebagai contoh adalah dalam pengembangan aplikasi diharuskan fitur autentikasi dikembangkan terlebih dahulu sebelum fitur kuis. Pada fitur kuis terdapat kebutuhan untuk mencatat progres dari *user* oleh karena itu fitur autentikasi perlu dikembangkan terlebih dahulu sebelum fitur kuis untuk menyediakan data *user* yang dibutuhkan. Fitur *timeline* pada Jira memungkinkan untuk memvisualisasikan *dependency* antar *task* seperti pada Gambar 3.4. Dengan ini, *project manager* dapat menentukan prioritas pada setiap *backlog* dan memberikan gambaran pada anggota tim mengenai alur pengembangan.



Gambar 3.3 Timeline Jira

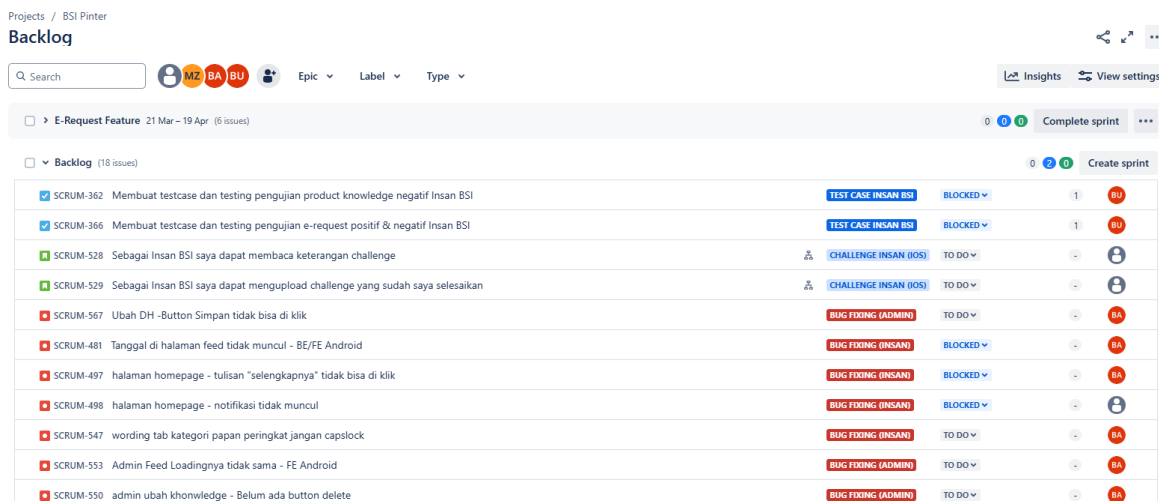


Gambar 3.4 Dependency antar item pada timeline

Sesuai pada *lifecycle* Scrum pada Bab 2, *product owner* menentukan daftar *product backlog* yang akan dieksekusi pada Sprint. Jira memiliki fitur *backlog* yang digunakan untuk mengelola *item product backlog* yang akan didistribusikan pada seluruh alur pengembangan, dapat dilihat pada Gambar 3.5. Pada pengembangan aplikasi ini *product owner* memanfaatkan fitur tersebut untuk mengelola seluruh *product backlog* yang dibutuhkan. Fitur ini berisi seluruh *product backlog* dalam berbagai macam tipe. Pada aplikasi ini *product backlog* umumnya terbagi kedalam beberapa tipe yaitu *user story*, *task*, dan *bug*. Beberapa *product*

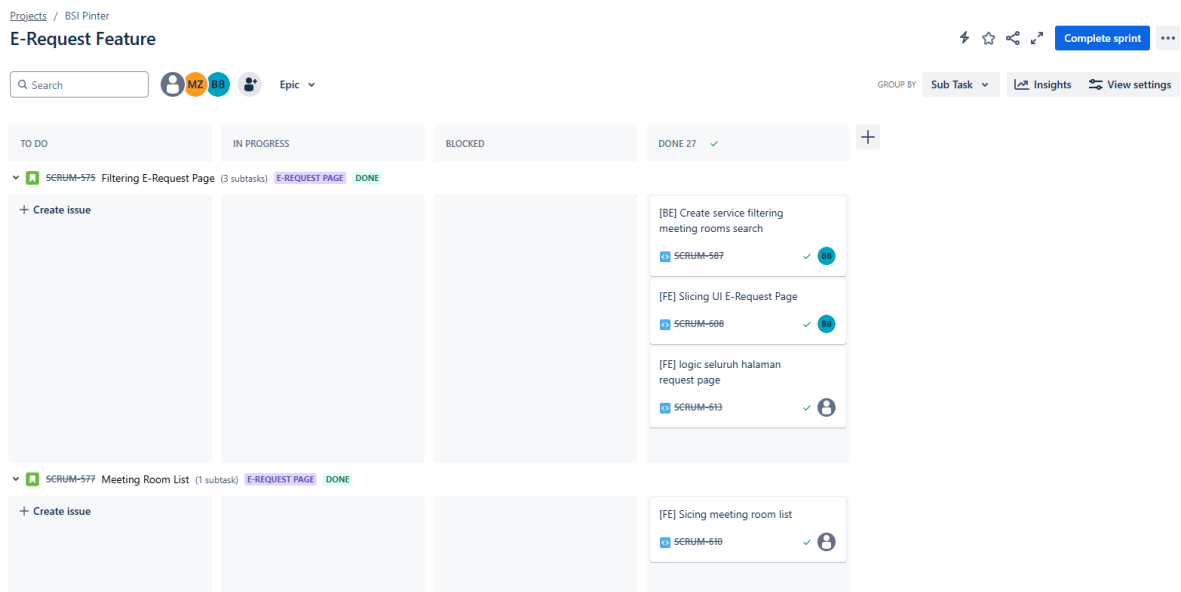
backlog akan dipilih untuk dimasukkan kedalam Sprint Backlog untuk dialokasikan pada pelaksanaan Sprint. Sprint Backlog merupakan kumpulan *backlog* yang diambil dari daftar *product backlog* utama untuk dikerjakan pada Sprint. Proses pengumpulan *product backlog* menjadi Sprint Backlog dilakukan pada fase Sprint Planning Meeting.

Seperti yang telah disebutkan sebelumnya, terdapat beberapa tipe *backlog* pada proyek ini. *Backlog* dengan tipe *user story* (warna hijau) merupakan deskripsi fungsi atau fitur atau fungsi tertentu yang merepresentasikan kebutuhan *user*. *Backlog* dengan tipe *task* (warna biru) merupakan *sub task* dari *user story*. Berdasarkan deskripsi kebutuhan pada *user story*, tim melakukan perincian *task* kedalam bentuk *sub task* yang merupakan langkah tindak lanjut untuk merealisasikan kebutuhan dari *user story*. Pada *board* terdapat *backlog* dengan tipe *bug* (warna merah) merupakan *bug* yang muncul pada sistem selama proses pengembangan atau setelah fase pengujian. Pada proyek ini, *backlog bug* ditetapkan oleh tim QA berdasarkan *test script* yang telah dibuat.



Gambar 3.5 Backlog Jira pada BSI PINTER

Pada Gambar 3.5 dapat dilihat contoh sampel *product backlog* pada proyek BSI PINTER. Sebelum fase Sprint dimulai, tim melakukan *Sprint Planning Meeting* untuk menentukan *Sprint Backlog* dari daftar *Product Backlog* utama. Setelah melakukan *Sprint planning meeting*, fase Sprint akan dimulai dengan fokus pada pengerjaan fitur sesuai pada Sprint Backlog. Seluruh *backlog* yang berupa *user story* dan *task* akan terlihat pada fitur Board pada Jira seperti pada Gambar 3.6.



Gambar 3.6 Jira Board

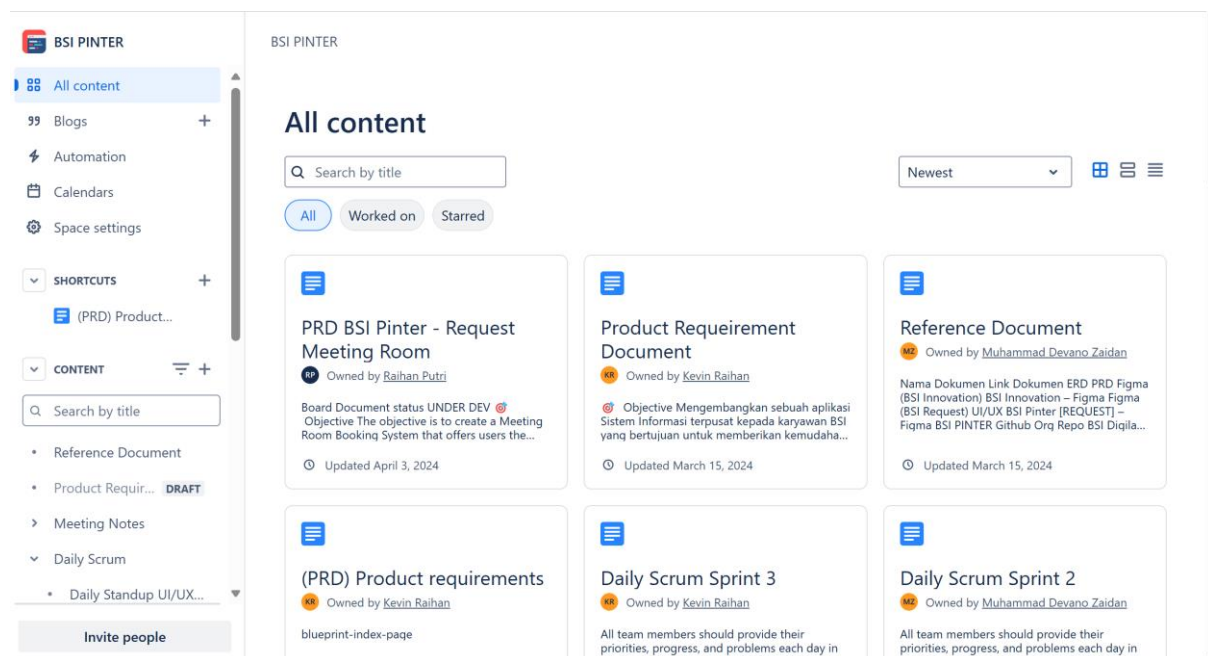
Pada proyek ini tim menggunakan Scrum Board yang dapat memvisualisasikan pekerjaan dalam lingkup yang lebih detail. Dalam *board* ini terdapat empat kolom yang menggambarkan status dari setiap pekerjaan dari tahap awal samapai selesai dikerjakan. Setiap tugas akan dipindahkan dari satu kolom ke kolom lainnya dengan status yang sesuai. Kolom *Todo* merupakan wadah untuk pekerjaan yang masih direncanakan, tetapi belum dimulai. Kolom *in progress* merupakan status yang menandai bahwa suatu pekerjaan sedang dilakukan. Kolom *blocked* merupakan status yang menandakan suatu pekerjaan sedang mengalami hambatan atau kendala sehingga belum bisa dilanjutkan. Kolom terakhir *Done* menandakan bahwa suatu pekerjaan telah selesai. Dalam konteks pengembangan aplikasi *backend*, maka fitur yang sesuai *user story* telah selesai dikembangkan dan siap untuk dilakukan pengujian. *User story* yang telah selesai berupa beberapa *endpoint* yang akan diintegrasikan dengan aplikasi *frontend*. Fitur yang telah selesai kemudian dilakukan pengujian oleh tim QA. Apabila fitur yang diuji terdapat *bug* atau tidak sesuai dengan *test script* maka tim QA akan menetapkan *backlog* dengan tipe *bug* pada *backlog* sesuai dengan persetujuan *product manager*.

Penggunaan Aplikasi Confluence

Pada pengembangan proyek ini, tim menggunakan aplikasi Confluence untuk melakukan dokumentasi terorganisir pada pengembangan aplikasi. Confluence dedesain khusus untuk memfasilitasi dokumentasi pengembangan proyek yang terintegrasi dengan Jira. Confluence memiliki beberapa fitur yang mendukung pendokumentasian proyek seperti fitur *pages* yang

digunakan tim untuk melakukan pencatatan dokumen teknis, catatan rapat, dan dokumen yang dibutuhkan untuk pendokumentasian Scrum.

Dalam pengembangan proyek ini, penggunaan Confluence sangat penting untuk memastikan semua informasi proyek terdokumentasi dengan baik dan mudah diakses oleh setiap anggota tim. Tim menggunakan fitur *pages* yang memungkinkan untuk menyusun dan mengelola dokumen terkait proyek secara terstruktur dengan hierarki yang jelas. Pada praktiknya, tim menggunakan Confluence untuk menyusun beberapa dokumen seperti Product Requirement Document (PRD), *daily scrum document*, *meeting notes* dan dokumen teknis pengembangan aplikasi. Seluruh dokumen pada proyek ini didokumentasikan dengan Confluence seperti pada Gambar 3.7.



Gambar 3.7 Confluence pada proyek BSI PINTER

3.3 Pelaksanaan Proyek BSI PINTER

Sub bab ini menjelaskan mengenai pelaksanaan proyek BSI PINTER dalam lingkup pengembangan aplikasi *backend*. Pada proyek ini penulis memiliki lingkup tanggungjawab dalam mengembangkan aplikasi BSI PINTER pada sisi *backend*. Pengembangan aplikasi *backend* didasarkan pada penggunaan metode Scrum secara keseluruhan, sebagaimana yang telah dijelaskan pada Bab 3.2 tentang penerapan metode Scrum. Pada Bab 3.3.1 diberikan gambaran umum mengenai keseluruhan fitur yang dikerjakan oleh penulis. Sebelum masuk pada penjelasan untuk setiap *endpoint*, pada Bab 3.3.2 dijelaskan mengenai arsitektur yang

digunakan pada BSI PINTER, menunjukkan alur request dan komponen apa saja yang mendukung implementasi fitur. Dalam setiap implementasi fitur dijelaskan pada Bab 3.3.3 mengenai alur pembuatan fitur menerjemahkan requirement menjadi sebuah fitur. Bab 3.3.4 sampai Bab 3.3.7 menjelaskan implementasi fitur pada setiap *endpoint* pada aplikasi *backend* BSI PINTER.

3.3.1 Deskripsi Project

Aplikasi BSI PINTER dikembangkan untuk memenuhi kebutuhan internal PT Bank Syariah Indonesia dalam hal pengelolaan informasi dan operasional karyawan. Aplikasi ini dirancang sebagai solusi terpadu yang mengintegrasikan beberapa fitur penting seperti pengelolaan kuis, autentikasi, registrasi, dan pengelolaan ruang *meeting*. Lingkup pengembangan aplikasi *backend* mencakup pengembangan *backend server* yang menangani logika bisnis, pengelolaan data, dan penyediaan API yang dapat diakses oleh aplikasi *frontend*.

Secara umum, sesuai pada dokumen perencanaan proyek aplikasi BSI PINTER direncanakan memiliki fitur berikut:

1. Materi Product Knowledge

BSI PINTER menyediakan daftar Materi Product Knowledge yang dapat dibaca oleh seluruh insan BSI. Product knowledge berisi materi mengenai produk-produk yang dimiliki oleh Bank BSI.

2. Sistem Event dan Seminar

Pihak manajemen dapat menggunakan BSI PINTER untuk memberikan informasi mengenai event dan seminar yang akan diselenggarakan oleh Bank BSI.

3. Kuis, Challenge, dan Rewards Seputar Bank BSI

BSI PINTER memberikan evaluasi pemahaman materi melalui kuis. Terdapat fitur challenge/tantangan yang dapat dikerjakan oleh insan BSI untuk mendapatkan rewards berdasarkan poin yang diperoleh. Hasil pengerjaan kuis dan tantangan menghasilkan poin yang dapat dikumpulkan untuk memperoleh *rewards*.

4. Beranda Informasi Terbaru dari manajemen

Pihak Manajemen dapat memberikan informasi terupdate kepada insan BSI, memberikan informasi penting dan terbaru oleh pihak manajemen.

5. Reporting Presensi Quiz dan Seminar

Pihak manajemen dapat melakukan pengecekan hasil pengerjaan kuis dan kedatangan seminar pada setiap anggota departemennya. Pihak Manajemen dapat melakukan

pengecekan pada anggota yang sudah dan belum mengerjakan atau mendatangi *event* pada seminar.

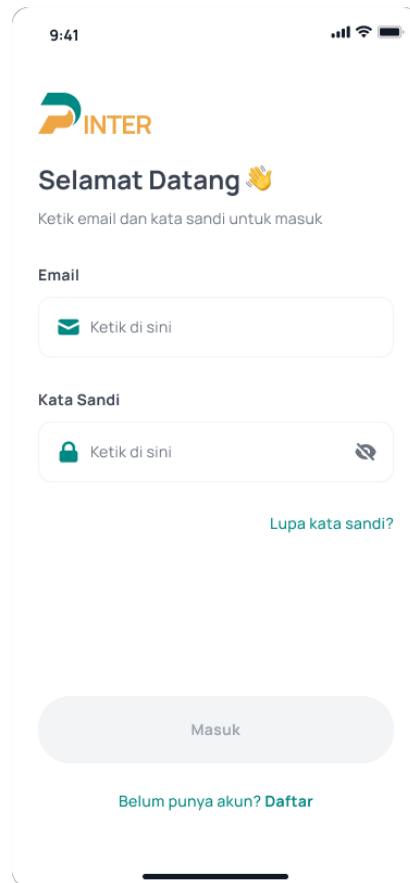
Sesuai pada fitur diatas, penulis mendapatkan bagian untuk melakukan implementasi fitur autentikasi dan registrasi, fitur kuis dan request ruang *meeting*. Fitur yang dikerjakan oleh penulis ditentukan berdasarkan proses diskusi dengan mentor dan *product manager* pada fase Sprint Planning dengan memperhatikan kompetensi anggota dan kompleksitas *backlog*. Pada fitur autentikasi dan registrasi, penulis mendapat *assignment* oleh Product Owner. Pada fitur kuis, penulis melakukan inisiatif pengambilan *backlog* sesudah berdiskusi dengan *product owner* dan anggota tim lainnya. Pada fitur pengelolaan ruang *meeting*, penulis mendapatkan *assignment* oleh Product Owner.

Fitur autentikasi dan registrasi

Tabel 3. 1 Backlog fitur autentikasi dan registrasi

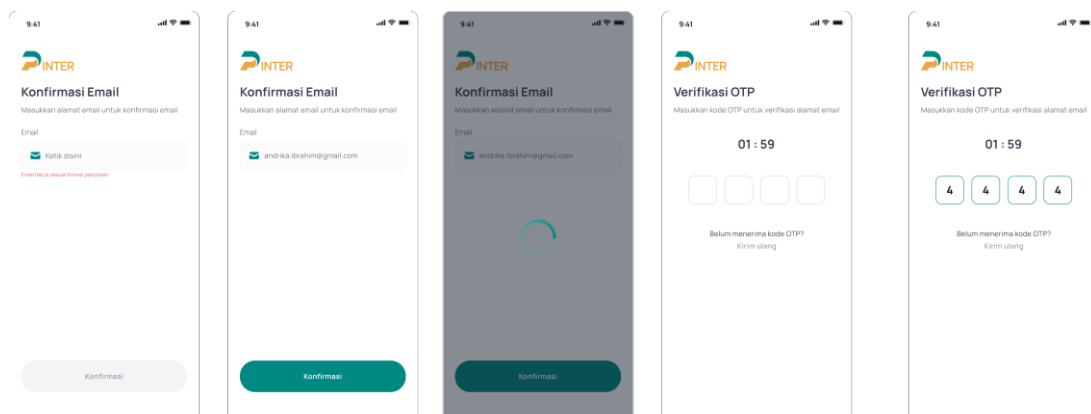
Backlog	Acceptance Criteria
Sebagia insan BSI saya dapat melakukan registrasi kedalam aplikasi BSI PINTER	Terdapat verifikasi <i>Email</i> menggunakan OTP
	Waktu countdown OTP selama dua menit
	Terdapat Countdown saat mengirimkan OTP
	Melakukan input data diri dengan mengirimkan nama lengkap, NIP, jabatan, kantor cabang, <i>password</i> , dan konfirmasi <i>password</i>
Sebagai insan BSI saya dapat melakukan autentikasi pada aplikasi BSI PINTER	Memasukkan kredensial dengan <i>username</i> dan <i>password</i>
	Terdapat validasi <i>email</i>
	Terdapat validasi untuk <i>password</i> minimal delapan karakter
	Mekanisme <i>login</i> menggunakan JWT

Tabel 3. 1 menunjukkan backlog yang diambil oleh penulis untuk melakukan implementasi fitur autentikasi dan registrasi. Fitur tersebut memiliki dua *backlog, user story* menggambarkan bahwa *user* dapat melakukan registrasi dan autentikasi pada aplikasi BSI PINTER. Pada fitur registrasi, user diharuskan melakukan verifikasi *email* menggunakan OTP, serta menginputkan data diri sesuai *acceptance criteria* yang telah didefinisikan. Pada bagian autentikasi, sistem diharuskan untuk menyediakan mekanisme autentikasi menggunakan JWT.



Gambar 3.8 Halaman Autentikasi

Langkah awal pada alur fitur ini adalah menampilkan halaman *login* bagi *user* untuk melakukan autentikasi seperti pada Gambar 3.8. Apabila *user* belum memiliki akun maka *user* dapat melakukan registrasi dengan menekan tombol daftar. Langkah pertama pada proses registrasi, *user* melakukan verifikasi *email* melalui OTP. *User* memasukkan *email* lalu menekan tombol konfirmasi untuk mendapatkan kode OTP. Setelah mendapatkan OTP, *user* melakukan input OTP untuk melakukan verifikasi kode.



Gambar 3.9 Halaman Verifikasi OTP

Gambar 3. 10 Halaman pendaftaran akun

Setelah melakukan proses verifikasi OTP, *user* melanjutkan proses registrasi dengan melakukan pengisian data diri sesuai informasi yang diminta pada halaman pendaftaran akun seperti pada Gambar 3. 10. Data yang perlu diisi meliputi Nama Lengkap, NIP, Regional, Kantor Cabang/Departemen, Jabatan, serta kata sandi yang akan digunakan. Setelah semua data terisi dengan benar, pengguna dapat melanjutkan dengan menekan tombol 'Simpan' untuk menyelesaikan proses pendaftaran akun. Setelah menyelesaikan proses pendaftaran akun pengguna dapat melakukan autentikasi dengan melanjutkan pada halaman login seperti pada Gambar 3.8.

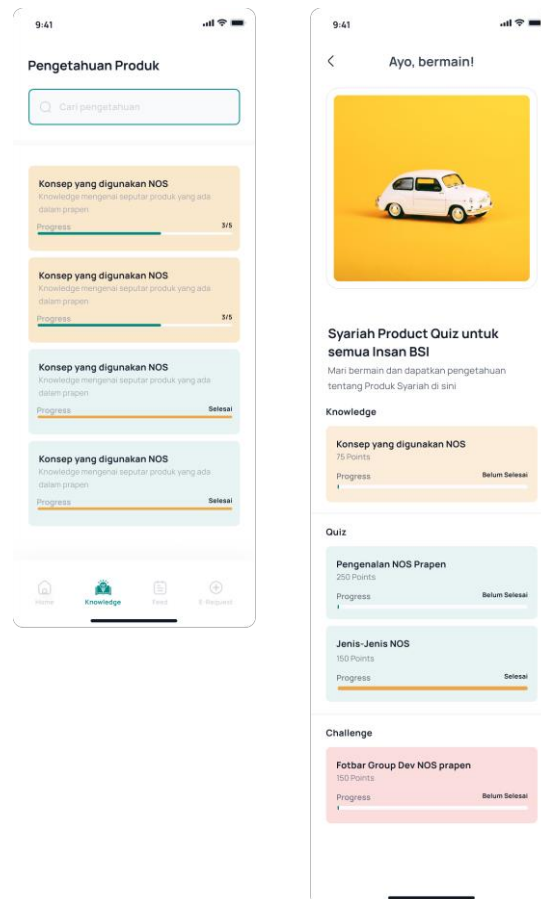
Fitur Kuis

Tabel 3. 2 Backlog fitur kuis *user*

Backlog	Acceptance criteria
Sebagai insan BSI saya dapat mengerjakan kuis	Menampilkan Kuis dalam dua tipe, pilihan ganda dan true false
	Menampilkan hasil jawaban setelah check answer
	Terdapat jumlah soal atau halaman pada soal kuis
	Jika soal memiliki jumlah karakter panjang maka container akan menyesuaikan dengan panjang soal

Sebagai insan BSI Saya dapat membagikan hasil kuis yang telah dikerjakan	Kuis yang dikerjakan dapat dibagikan kedalam <i>feeds</i>
	Terdapat opsi untuk membagikan ke semua insan BSI
Sebagai insan BSI saya akan mendapatkan poin ketika sudah menyelesaikan kuis	Jika insan menyelesaikan kuis maka poin yang dimiliki oleh insan akan bertambah sesuai poin yang tertera
Sebagai insan BSI saya dapat melihat hasil kuis setelah menyelesaikan kuis	Menampilkan total jawaban benar dan salah
	Menampilkan total poin yang didapatkan
	Menampilkan hasil kunci jawaban

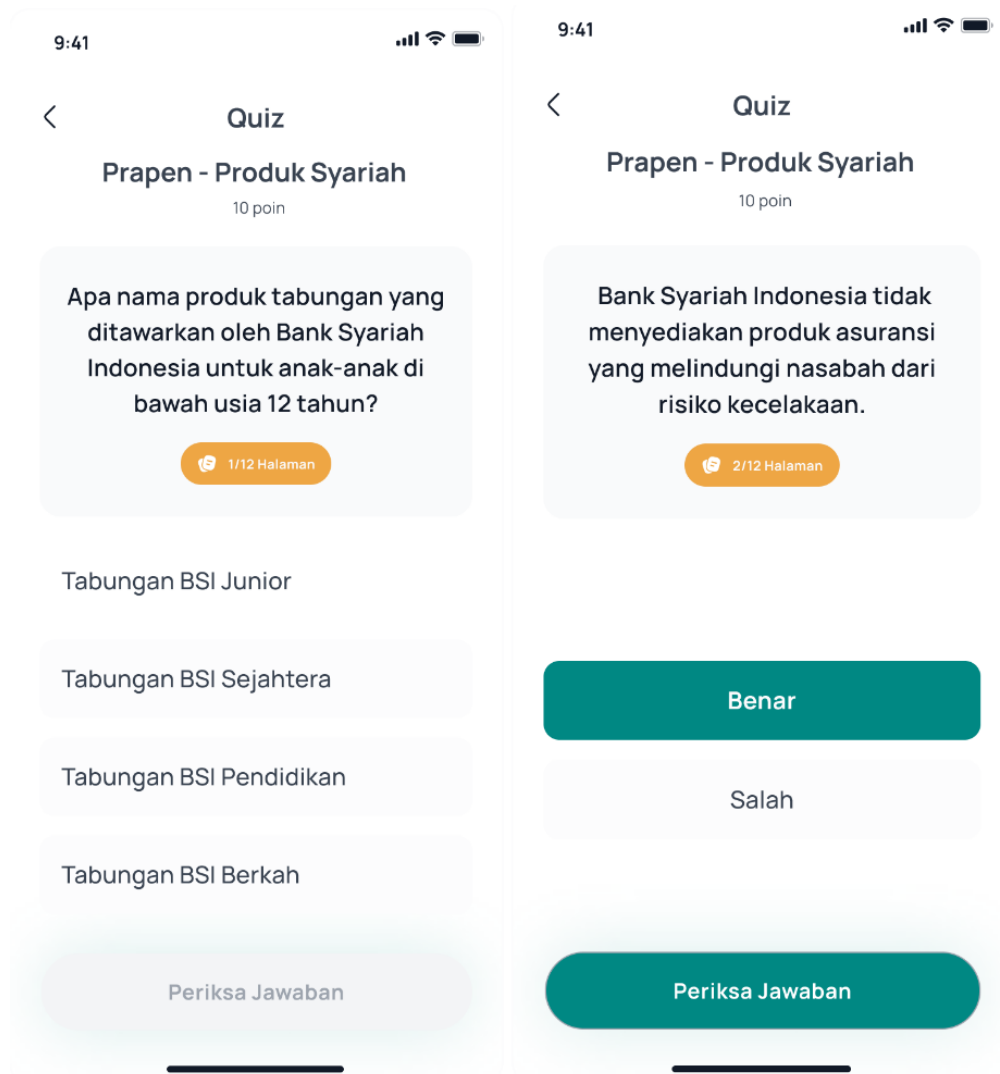
Tabel 3. 2 menunjukkan backlog yang diambil oleh penulis untuk melakukan implementasi fitur kuis pada sisi *user*. Fitur kuis pada sisi *user* memiliki empat *user story* beserta acceptance criterianya. *User story* menggambarkan kebutuhan *user* untuk dapat mengerjakan soal kuis dan membagikan hasil kuis yang telah dikerjakan. Hasil kuis yang dikerjakan dibagikan dalam bentuk *feeds*.



Gambar 3. 11 Halaman product knowledge

Product knowledge berisi mengenai materi dan artikel mengenai produk-produk BSI. Untuk membuka halaman product knowledge, *user* menekan tombol pada navbar dengan ikon product knowledge seperti pada Gambar 3. 11. Halaman akan menampilkan daftar product

knowledge beserta progres dari *user*. *User* kemudian memilih product knowledge untuk kemudian melihat detail product knowledge yang berisi knowledge (artikel produk), quiz, dan challenge. Untuk mengerjakan sebuah kuis, *user* memilih pada bagian Quiz untuk mendapatkan detail kuis.



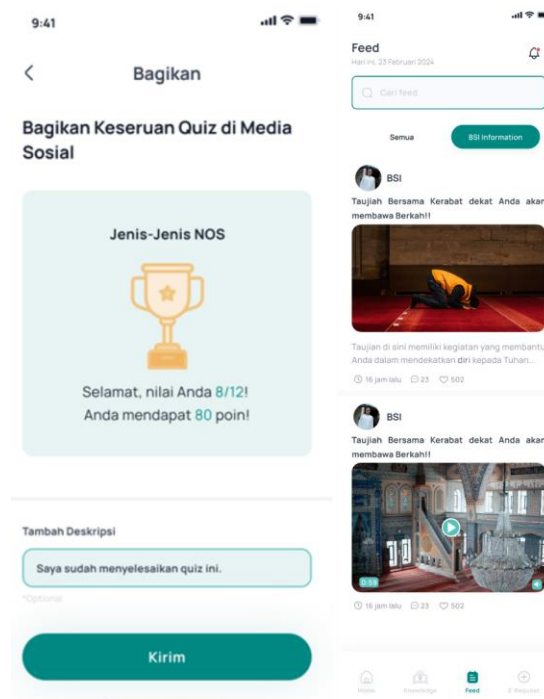
Gambar 3. 12 Halaman detail kuis

Setelah memilih bagian Quiz pada product knowledge, *user* akan mendapatkan detail kuis untuk dikerjakan seperti pada Gambar 3. 12. Terdapat dua jenis soal yang tersedia yaitu pilihan ganda dan true/false. *User* mengerjakan soal secara berurutan, *user* dapat keluar dari halaman dan melanjutkan pada soal yang belum diselesaikan. Setelah selesai mengerjakan seluruh soal *user* akan mendapatkan hasil kuis seperti pada Gambar 3. 13. Hasil kuis menampilkan hasil poin yang didapat serta kunci jawaban pada setiap soal.



Gambar 3. 13 Halaman hasil kuis

Kemudian *user* menekan tombol selesai untuk menyelesaikan kuis untuk kemudian membagikan hasil kuis seperti pada Gambar 3. 14 (gambar pertama dari kiri). Hasil kuis dibagikan pada halaman *feeds* seperti pada Gambar 3. 14 (gambar kedua dari kiri). *Feeds* berisi cerita pengguna aplikasi BSI PINTER termasuk hasil kuis yang telah dikerjakan, hasil pengerjaan tantangan.

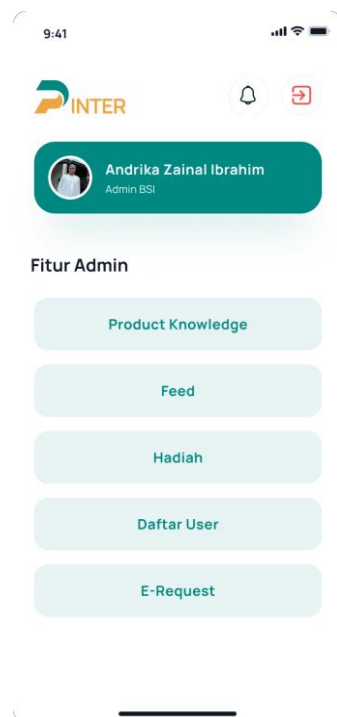


Gambar 3. 14 Halaman membagi hasil kuis kedalam *feeds*

Tabel 3. 3 Backlog fitur kuis admin

Backlog	Acceptance criteria
Sebagai admin saya dapat menambahkan soal kuis yang dikerjakan oleh insan BSI	Dapat menambahkan judul kuis dan soal kuis
	Dapat Menambahkan kuis sesuai <i>product knowledge</i>
	Terdapat pilihan untuk jenis soal kuis
	Soal terdiri atas pilihan ganda dan true/false
Sebagai admin saya dapat mengubah data soal kuis yang dikerjakan oleh insan BSI	-
Sebagai admin saya dapat menghapus soal kuis yang dikerjakan oleh insan BSI	Jika kuis dihapus soal yang berelasi juga terhapus

Tabel 3. 3 menunjukkan backlog pada fitur kuis sisi admin. Terdapat tiga *user story* yang menggambarkan kebutuhan pada sisi admin untuk dapat menambahkan soal pada kuis, mengubah data pada kuis, dan menghapus soal pada kuis.



Gambar 3. 15 Halaman utama admin

Gambar 3. 15 merupakan halaman utama admin yang terdiri dari beberapa fitur yang dapat diakses oleh admin untuk mengelola fitur pada aplikasi. Untuk mengakses fitur kuis, admin dapat membuka pada bagian *product knowledge*. Selanjutnya akan ditampilkan detail *product knowledge* seperti pada Gambar 3. 16.

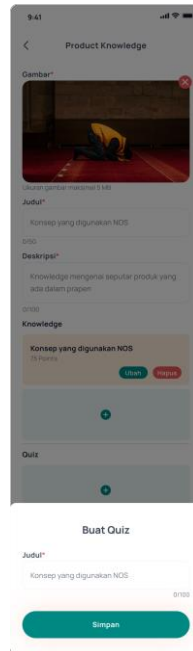
Gambar 3. 16 Halaman detail product knowledge admin

Pada halaman detail *product knowledge* seperti pada Gambar 3. 16 terdapat bagian kuis untuk admin melakukan fungsi admin kuis seperti menambahkan, mengubah, menghapus, dan mengedit kuis. Untuk menghapus kuis, admin menekan tombol hapus berwarna merah. Untuk mengubah data kuis, admin dapat menekan tombol ubah pada bagian kuis.

Gambar 3. 17 Halaman admin untuk menambahkan kuis

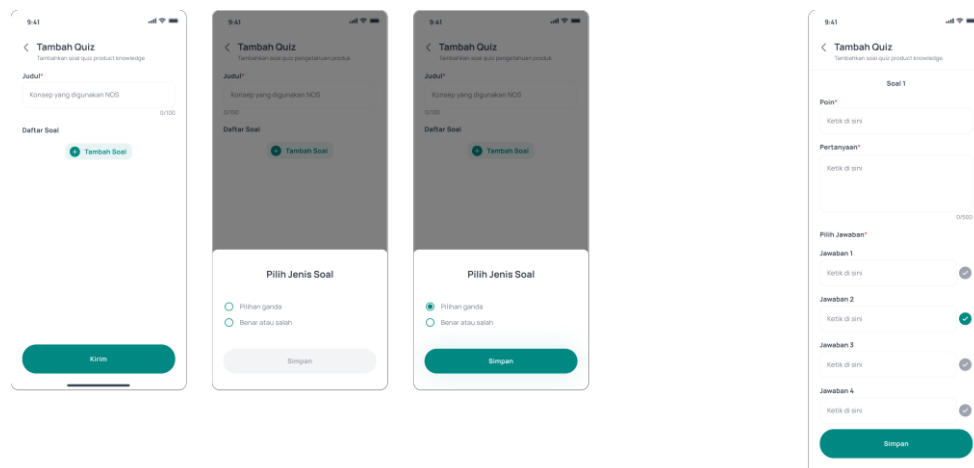
Untuk menambahkan kuis pengguna menekan tombol tambah bagian kuis pada kolom terakhir (dengan ikon tambah) seperti pada Gambar 3. 16. Kemudian akan ditampilkan

halaman untuk menambah judul kuis terlebih dahulu pada *bottom sheets* seperti pada Gambar 3. 18. Langkah awal dalam menambahkan data kuis adalah dengan menambahkan judul kuis terlebih dahulu seperti pada Gambar 3. 18.



Gambar 3. 18 Bottom sheets menambahkan judul kuis

Setelah menambahkan judul pada kuis, maka otomatis sistem akan menambahkan data kuis yang masih kosong. Setelah menambahkan judul pada kuis, admin kemudian dapat menambahkan beberapa soal pada kuis seperti pada Gambar 3. 19.



Gambar 3. 19 Alur halaman menambahkan soal kuis pada admin

Pada Gambar 3. 19, alur menambahkan soal pada kuis dimulai dari halaman pertama dari kiri. Admin menekan tombol tambah soal, kemudian sistem akan memunculkan sebuah *bottom sheets* untuk memilih jenis soal. Setelah memilih jenis soal maka langkah selanjutnya adalah mengisi detail soal kuis seperti pada halaman terakhir (nomor empat dari kiri). Admin

menambahkan detail soal kuis seperti poin, pertanyaan, dan kunci jawaban. Setelah mengisi data soal kuis, admin dapat menyimpan soal kemudian melakukan langkah serupa untuk menambahkan soal lainnya.

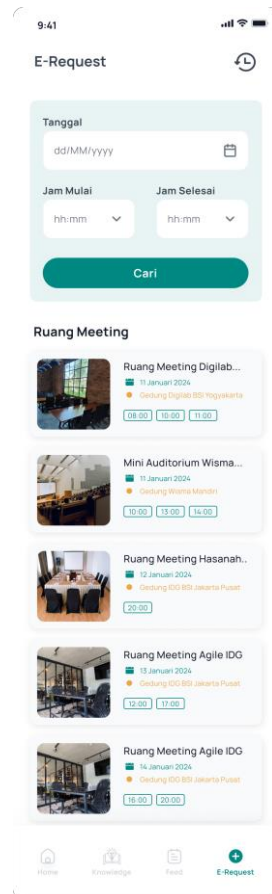
Fitur Pemesanan Ruang Meeting

Tabel 3. 4 Backlog fitur pemesanan ruang *meeting* sisi *user*

Backlog	Acceptance Criteria
Sebagai insan BSI saya dapat melihat daftar ruangan <i>meeting</i>	Menampilkan daftar ruang <i>meeting</i> yang tersedia sesuai dengan data daftar ruang <i>meeting</i> pada confluence
	Terdapat status penggunaan ruang <i>meeting</i> (digunakan/tersedia)
	Terdapat lokasi pada setiap ruang <i>meeting</i>
Sebagai insan BSI saya dapat melihat detail ruangan <i>meeting</i>	Terdapat status penggunaan ruang <i>meeting</i> (digunakan/tersedia)
	Terdapat jumlah kapasitas ruang <i>meeting</i>
Sebagai insan BSI saya dapat melakukan pemesanan ruang <i>meeting</i>	Terdapat daftar fasilitas ruang <i>meeting</i>
	Dapat memilih tanggal pemesanan
Sebagai insan BSI saya dapat melihat riwayat pemesanan ruang <i>meeting</i>	Dapat memilih jam mulai dan jam berakhir
	-

Tabel 3. 4 menunjukkan backlog pada fitur request ruang *meeting* pada sisi *user*. Terdapat empat backlog utama yang menggambarkan kebutuhan untuk melihat daftar ruang *meeting*, mendapatkan data detail ruang *meeting*, melakukan pemesanan ruang *meeting*, dan mendapatkan data riwayat pemesanan ruang *meeting* pada insan BSI.

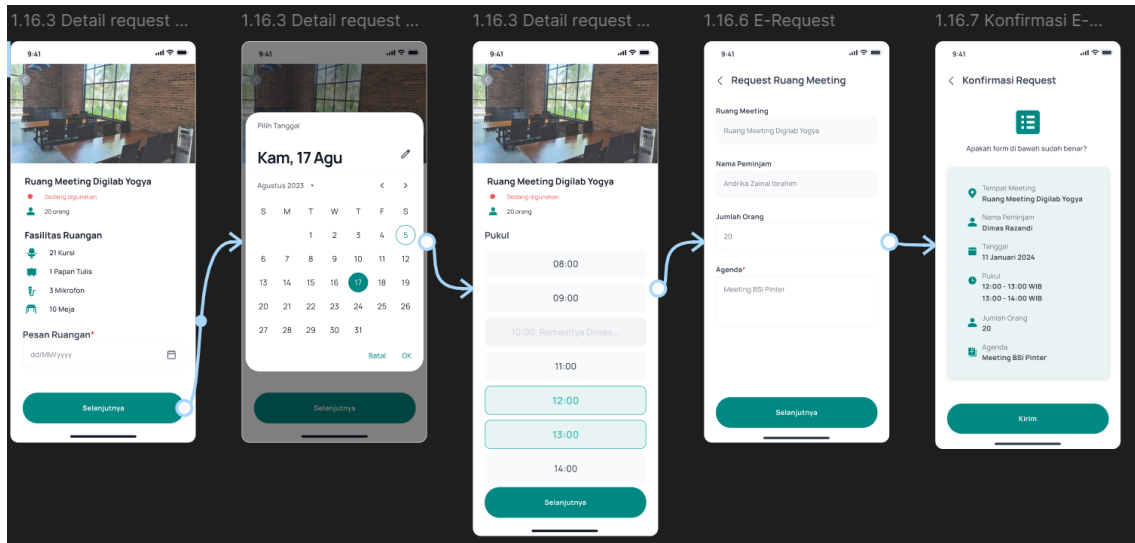
Seperti pada Gambar 3. 20, halaman utama ruang *meeting* terdiri dari daftar ruang *meeting* yang dapat digunakan oleh insan BSI. Pada setiap item, terdapat status ketersediaan pada ruang *meeting*. Admin dapat melihat detail ruang *meeting* dengan menekan pada item pada daftar ruang *meeting*. Detail ruang *meeting* ditampilkan seperti pada Gambar 3. 21, menampilkan detail ruangan seperti kapasitas, ketersediaan, dan daftar fasilitas.



Gambar 3. 20 Halaman utama *request* ruang *meeting*

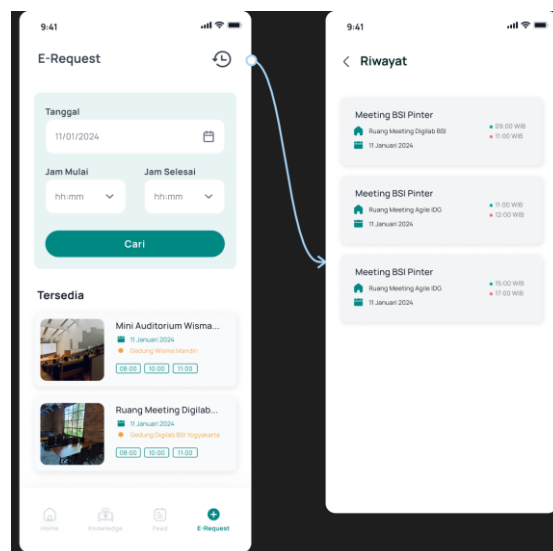


Gambar 3. 21 Halaman detail ruang *meeting*



Gambar 3. 22 Alur halaman pemesanan ruang *meeting*

Seperti pada Gambar 3. 22, untuk melakukan pemesanan ruang *meeting* insan BSI melakukan pemesanan melalui halaman detail ruang *meeting* (pertama dari kiri) kemudian mengisi tanggal pemesanan yang akan dilakukan. Setelah mengisi tanggal pemesanan, insan BSI kemudian mengisi jam mulai dan berakhirnya penggunaan ruang *meeting*. Langkah terakhir adalah mengisi data detail ruang *meeting* (nomor empat dari kiri) yaitu jumlah orang, dan agenda ruang *meeting*. Kolom ruang *meeting* dan nama peminjam otomatis terisi secara otomatis oleh sistem android.



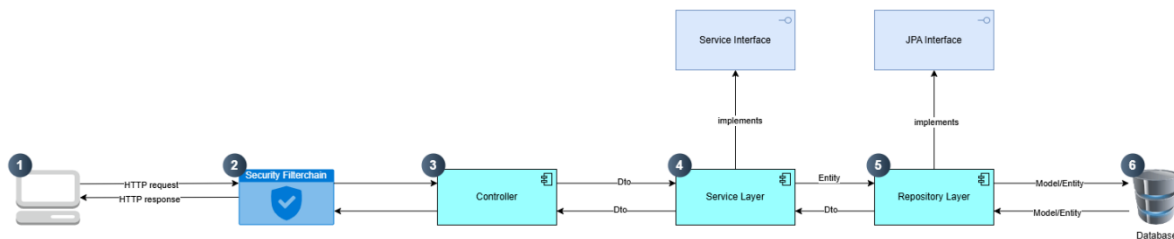
Gambar 3. 23 Halaman riwayat pemesanan ruang *meeting*

Seperti pada gambar 3.2, insan BSI dapat melihat riwayat pemesan ruang *meeting* yang telah dilakukan. Daftar riwayat pemesanan ruang *meeting* menampilkan daftar item ruang

meeting yang telah dipesan dengan setiap item menampilkan data pemesanan yaitu agenda, nama ruangan, tanggal pemesanan, dan jam mulai dan berakhirnya *meeting*.

3.3.2 Arsitektur Aplikasi Backend

Aplikasi *Backend* BSI PINTER dikembangkan dengan menggunakan framework Spring. Aplikasi menggunakan arsitektur MVC (Model View Controller) yang bersifat *monolith*. Pada konteks pengembangan aplikasi ini, View tidak secara langsung didefinisikan sebagai antarmuka atau UI (*User Interface*) seperti pada aplikasi *frontend*. Namun, *view* pada konteks REST API merupakan *response* yang dikirimkan oleh aplikasi *backend* dalam representasi data dengan format yang telah ditentukan. Pada aplikasi ini format yang digunakan adalah JSON (Javascript Object Notation).



Gambar 3. 24 Arsitektur Spring yang digunakan di BSI PINTER

Aplikasi *backend* BSI PINTER dikembangkan menggunakan Spring Boot dengan arsitektur monolitik MVC. Framework Spring Web digunakan untuk menyediakan layanan REST API dengan Apache Tomcat sebagai server bawaan. Berikut adalah alur arsitektur aplikasinya:

Arsitektur aplikasi *backend* BSI PINTER diimplementasikan seperti pada **Error! Reference source not found.**, secara umum terdapat tiga layer yang menyusun fungsionalitas keseluruhan aplikasi. Aplikasi ini menggunakan modul Spring Web untuk menyediakan fungsionalitas *web server* yang menyediakan interaksi dengan REST API antara *client* dan *server*. Dengan menggunakan Spring Web, Spring Boot otomatis menyediakan konfigurasi untuk membangun sebuah web server menggunakan Apache Tomcat. Arsitektur aplikasi ini mempunyai alur sebagai berikut:

1. Client

Client mengirimkan *request* HTTP ke server melalui *endpoint* yang disediakan. Spring Web memetakan path pada *request* ke controller yang sesuai.

2. Security Layer

Menggunakan Spring Security dengan JWT (JSON Web Token) untuk autentikasi dan otorisasi. JWT Filter memvalidasi token sebelum *request* diteruskan ke controller. *Endpoint* yang memerlukan autentikasi disaring oleh Security Filter Chain.

3. Controller Layer

Controller menerima *request* dari client, lalu meneruskannya ke Service Layer. Data yang dikirimkan oleh *client* dikonversi menjadi DTO (Data Transfer Object) untuk dikirim antar layer.

4. Service Layer

Berfungsi untuk menjalankan logika bisnis aplikasi. Service layer berinteraksi dengan Repository Layer untuk operasi *email*. Kelas service mengimplementasikan interface yang menyediakan metode-metode bisnis.

5. Repository Layer

Repository adalah interface yang mengimplementasikan JpaRepository untuk operasi CRUD. Layer ini menjembatani interaksi antara Service Layer dan *email*.

6. Database Layer

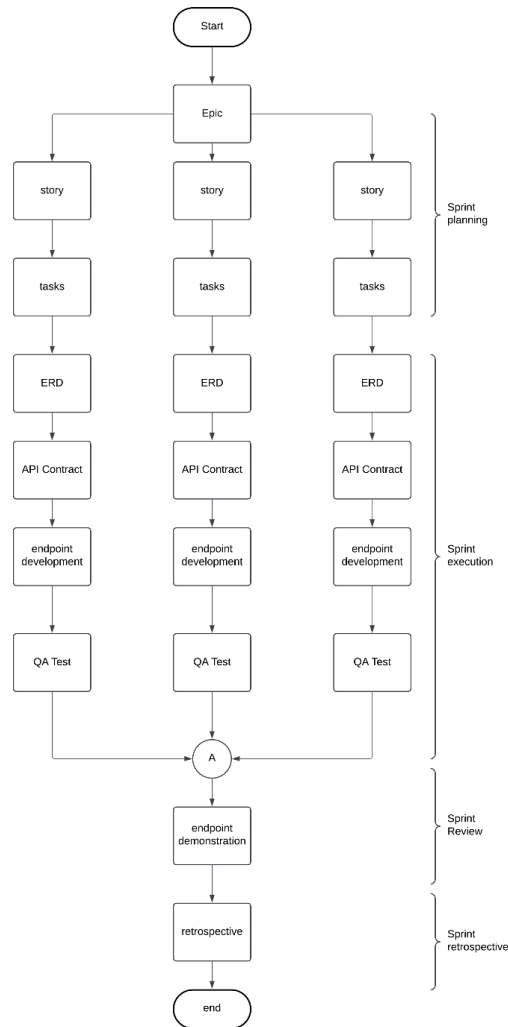
Layer ini bertanggung jawab menyimpan data secara persisten. Spring Data JPA digunakan sebagai abstraksi untuk operasi *email* dengan Hibernate ORM sebagai implementasi. Data disimpan dalam *email* melalui operasi CRUD.

3.3.3 Alur Pengembangan Backend

Untuk mengembangkan sebuah fitur pada aplikasi *backend* diperlukan sebuah proses yang terstruktur agar setiap komponen dapat terintegrasi dengan baik. Dalam konteks scrum alur tersebut merupakan rangkaian pengembangan yang diperlukan untuk mengeksekusi task-task pada *user stories* untuk merealisasikan fitur yang fungsional. Tahap tersebut dilakukan pada fase eksekusi sprint.

Alur implementasi *backend* digambarkan seperti pada diagram Gambar 3. 25. Alur tersebut merupakan rangkaian implementasi pada aplikasi *backend* dimulai dari sprint *planning* sampai dengan berakhirnya sprint. Pada sprint planning, ditentukan backlog yang akan dikerjakan yaitu Epic yang berisi beberapa *user stories*. Pada fase sprint planning penulis diharuskan melakukan *breakdown task* pada setiap *user story*. Setelah mendefinisikan task yang diperlukan, penulis dan anggota tim *backend* lainnya membuat rancangan ERD berdasarkan requirement pada *user story* dan desain UI. Dari ERD yang telah dibuat tim *backend* kemudian merancang API Contract yang dibutuhkan untuk menyediakan gambaran endpoint yang nantinya akan diintegrasikan oleh tim *backend*. Tim frontend dan tim *backend*

melakukan *meeting* untuk mendiskusikan *API Contract* yang telah dibuat. Tim *frontend* dan tim *backend* perlu memutuskan kesepakatan apakah *API Contract* telah disetujui bersama. Apabila hasil *API Contract* telah disepakati, maka tim *backend* kemudian mengembangkan aplikasi *backend* untuk menyediakan *endpoint* yang diperlukan untuk merealisasikan sebuah fitur. Setelah fitur telah selesai dikembangkan, fitur kemudian akan dilakukan pengujian lebih lanjut oleh tim QA berdasarkan skenario-skenario yang telah dirancang. Apabila fitur tidak lolos pengujian QA maka *backend* developer akan melakukan perbaikan terhadap backlog bug yang telah diberikan oleh QA. Jika fitur lolos pengujian, maka diakhir sprint setiap fitur yang telah selesai dikembangkan akan didemonstrasikan ke seluruh anggota tim scrum dan juga *stakeholder*. Fitur yang telah selesai ditambahkan dalam dokumentasi API dan siap untuk dilakukan integrasi oleh *frontend* pada Sprint Selanjutnya. Pada langkah terakhir, tim melakukan sprint *retrospective* untuk mengevaluasi pekerjaan scrum yang telah dilakukan dan mencari peluang perbaikan pada sprint selanjutnya.

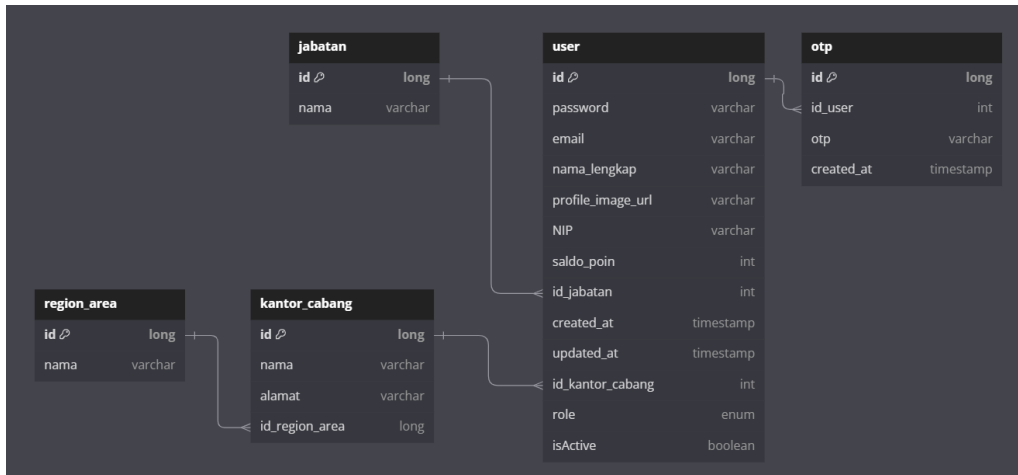


Gambar 3. 25 Alur pengembangan *backend*

3.3.4 Implementasi Fitur Autentikasi dan Registrasi

Sebelum *user* melakukan login untuk dilakukan autentikasi, *user* perlu melakukan proses registrasi sebagai langkah untuk mendapatkan kredensial yang sah. Setelah melakukan registrasi *user* mendapatkan kredensial yang sah, kredensial tersebut digunakan untuk melakukan proses *login* pada aplikasi. Pada proses *login*, sistem akan melakukan validasi kredensial. Jika kredensial valid maka kemudian sistem akan menghasilkan token JWT yang dapat digunakan untuk mengakses *endpoint* lainnya pada aplikasi *backend*.

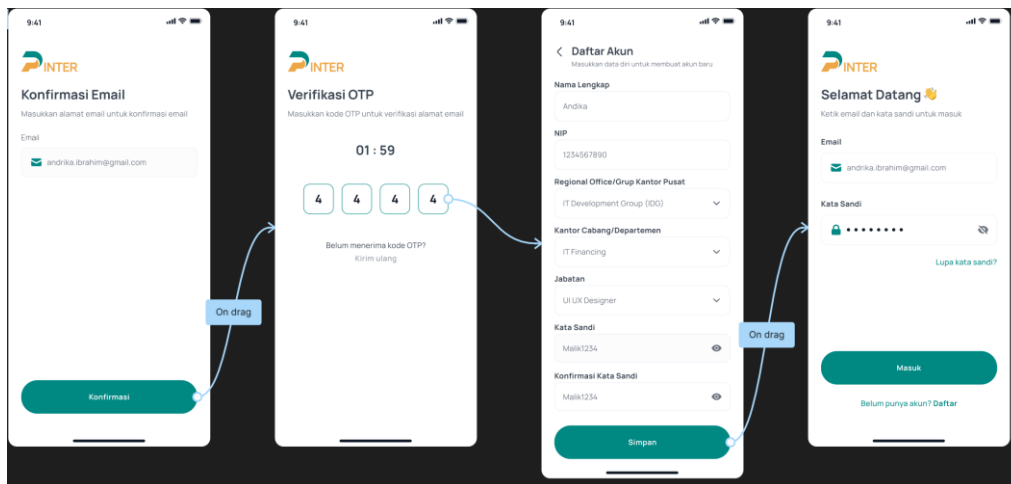
ERD



Gambar 3. 26 ERD untuk autentikasi dan registrasi

ERD pada Gambar 3. 26 menggambarkan kebutuhan entitas untuk menjalankan fungsionalitas pada fitur autentikasi dan registrasi. Terdapat lima entitas yaitu *user*, *otp*, *jabatan*, *kantor_cabang*, dan *region_area*, *user* merupakan entitas yang diperlukan untuk menyimpan data pegawai dan kredensial yang diperlukan untuk autentikasi. Sedangkan tabel *otp* digunakan untuk menyimpan data *otp* yang digunakan untuk *user* melakukan verifikasi pada proses registrasi. Tabel lainnya seperti *kantor_cabang* dan *region_area* digunakan untuk menyimpan informasi lokasi kerja insan BSI.

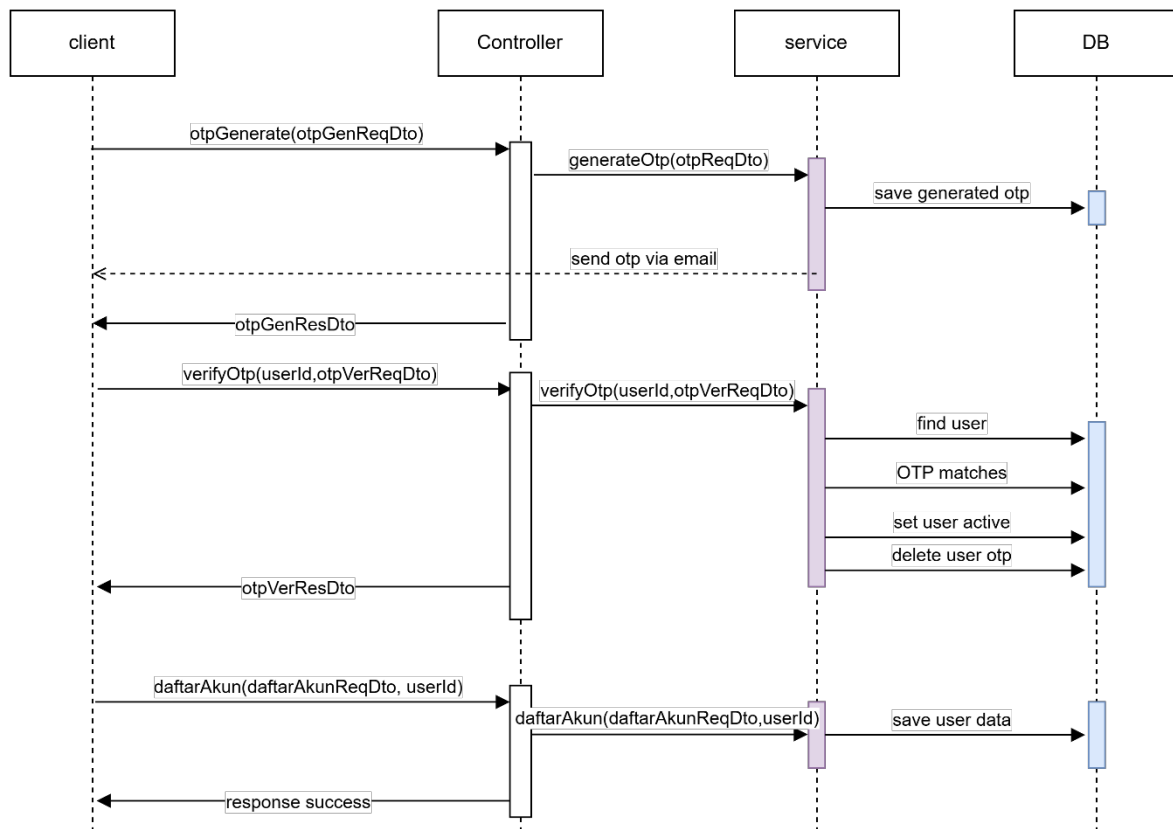
Registrasi



Gambar 3. 27 Alur registrasi

Proses registrasi digambarkan seperti alur pada Gambar 3. 27. Pertama *user* melakukan konfirmasi *email* dengan melakukan verifikasi OTP. *User* menerima kode OTP melalui *email* yang dikirimkan oleh sistem *backend*. Setelah menerima kode OTP melalui *email*, *user* melakukan verifikasi kode OTP yang telah diterima. Kemudian *user* mengisi data diri, *field*

email dan *password* merupakan kredensial *user* yang digunakan untuk melakukan proses login pada sistem.



Gambar 3. 28 Sequence diagram alur registrasi *backend*

Gambar 3. 28 merupakan sequence diagram yang menggambarkan alur registrasi pada sistem *backend* BSI PINTER. Terdapat tiga alur proses registrasi, alur pertama, *user* melakukan konfirmasi *email* untuk mendapatkan OTP melalui *request* pada *endpoint* /api/v1/auth/otp-generate dengan mengirimkan *request body* seperti pada Gambar 3. 29. Sistem akan memproses melalui OtpService pada method generateOtp() seperti pada Gambar 3. 30.

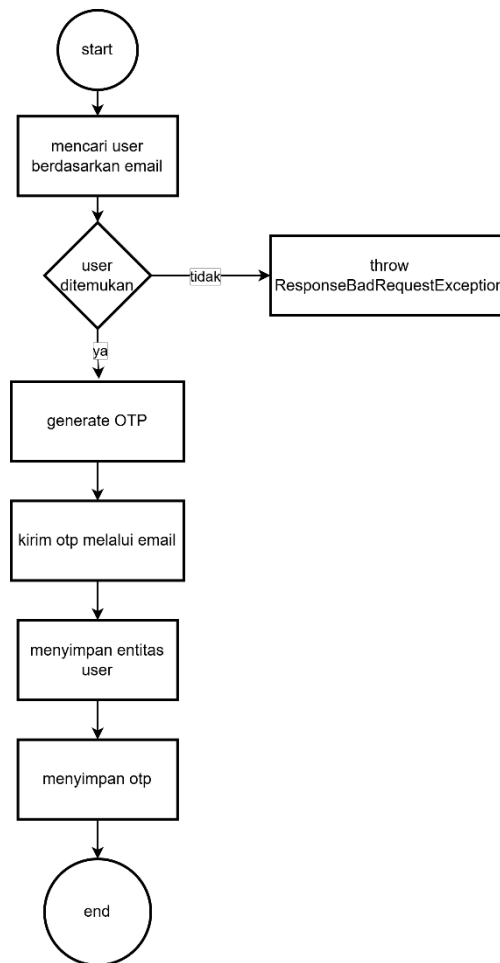
```

#endpoint
POST/api/v1/auth/otp-generate
#request
{"email": "sampleuser@gmail.com"}
#response
{
  "idUser" : 123,
  "email" : "sampleuser@gmail.com"
  "message": "kode OTP telah dikirim melalui email"
  "data": null
}
  
```

Gambar 3. 29 Request response generate OTP

Gambar 3. 30 menunjukkan proses penghasilan kode OTP pada sistem. Pertama, Sistem mencari entitas *user* berdasarkan *email*. Kemudian sistem menghasilkan kode OTP dengan

kombinasi yang telah ditentukan. Hasil OTP kemudian disimpan dalam tabel OTP lalu mengasosiasikannya dengan *user*.



Gambar 3. 30 Alur pemuatan kode OTP

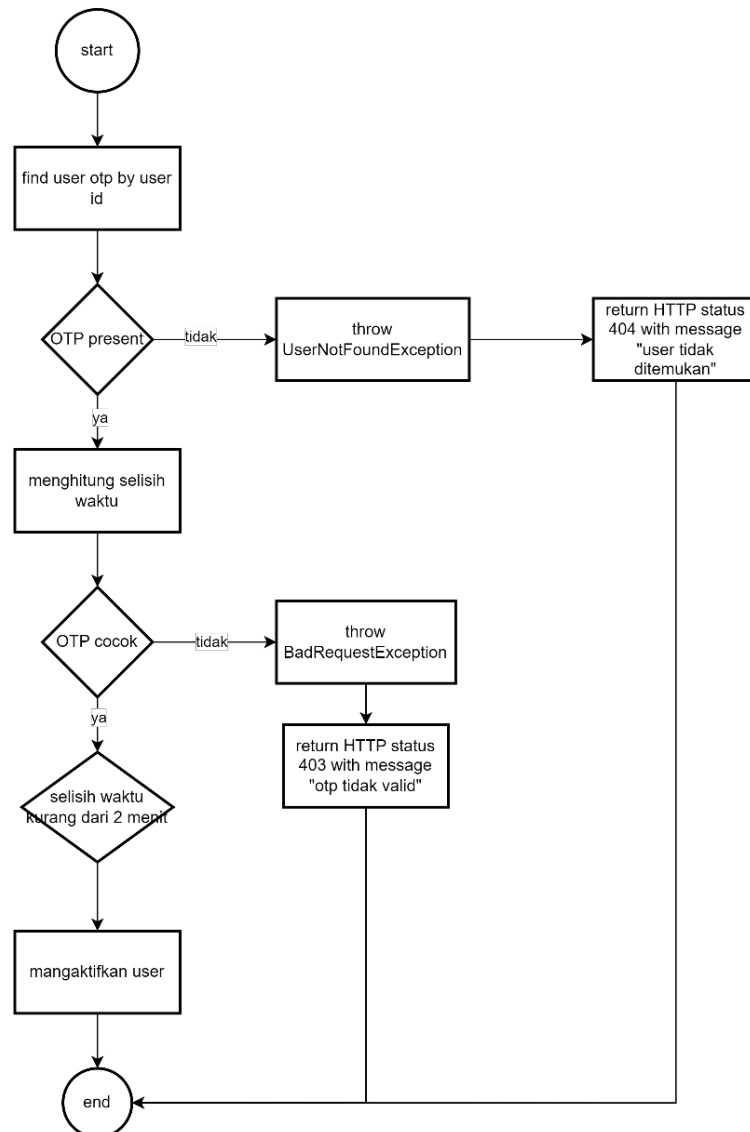
Langkah kedua, *user* melakukan konfirmasi OTP yang telah diterima. Client melakukan request pada *endpoint* `/api/v1/auth/{userid}/otp-verify` dengan mengirimkan *request* seperti pada Gambar 3. 31. Sistem akan memproses melalui *OtpService* seperti pada Gambar 3. 32.

```

#endpoint
POST /api/v1/auth/{userid}/otp-verify
#request
{
  "otp" : "8320"
}
#response
{
  "status" : 200,
  "message": "verifikasi OTP berhasil"
  "data" : null
}
  
```

Gambar 3. 31 Request response verifikasi OTP

Untuk melakukan verifikasi kode OTP yang dikirimkan oleh *user*, sistem mencocokkan OTP yang telah disimpan pada *email* dengan OTP *user* seperti pada Gambar 3. 32. Setelah proses verifikasi OTP berhasil, maka status *user* akan diaktifkan. Kemudian, *user* dapat melanjutkan ke langkah terakhir untuk mengisi data diri.



Gambar 3. 32 Alur verifikasi OTP

Langkah terakhir, *user* mengirimkan data diri untuk mengisi kredensial. *Client* melakukan *request* seperti pada Gambar 3. 33. *User* mengirimkan data-data kepegawaian yang diperlukan untuk melakukan registrasi pada aplikasi. Memastikan bahwa keseluruhan data telah sesuai dengan identitas pribadi.

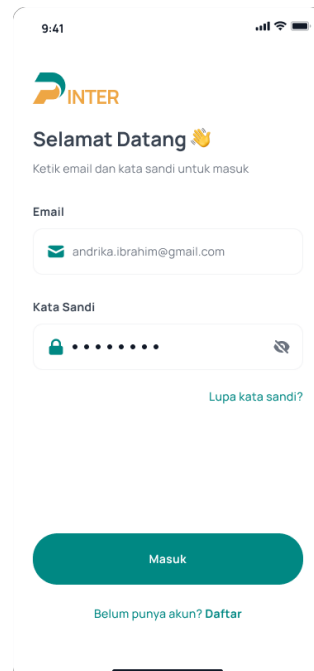
#endpoint POST api/v1/auth/16/otp-verification

```
#request
{
  "namaLengkap" : "Andrika",
  "nip" : "123456789",
  "jabatan" : 10,
  "kantorCabang" : 2,
  "password" : "password"
}
#response
{
  "status": 200,
  "message": "data berhasil disimpan",
  "data": {
    "name": "Andrika",
    "nip": 123456789,
    "jabatan": "staff",
    "kantorCabang": "IT Development"
  }
}
```

Gambar 3. 33 Request response pendaftaran akun

Setelah menyelesaikan proses registrasi, *user* telah menyimpan data diri dan mendapatkan kredensial yang valid. Kredensial tersebut kemudian digunakan untuk proses login pada aplikasi. Kredensial yang valid memastikan bahwa *user* dapat mengakses aplikasi sesuai role yang telah ditentukan.

Login dan Autentikasi



Gambar 3. 34 Halaman *login*

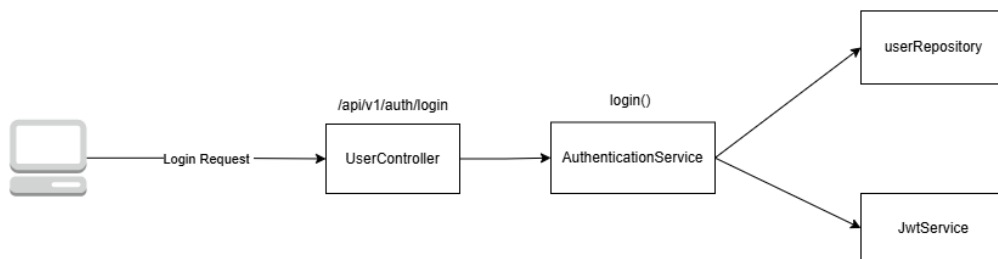
Untuk melakukan proses *login*, *user* mengisi kredensial pada halaman *login* seperti pada Gambar 3. 34. *Client* melakukan *request* seperti pada Gambar 3. 35, mengirimkan kredensial berupa *email* dan *password*. Sistem kemudian melakukan proses autentikasi dengan

memvalidasi kredensial *user*. Jika kredensial yang dikirimkan valid maka sistem akan menghasilkan token JWT yang dapat digunakan oleh *client* untuk mengakses *endpoint* lainnya sesuai role yang diberikan.

```
#endpoint
POST /api/v1/auth/login
#request
{
  "email": "devano@gmail.com",
  "password": "samplePassword"
}
#response
{
  "status": 200,
  "message": "login berhasil",
  "data": {
    "email": "devano@gmail.com",
    "role": "user",
    "token":
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeLqBgyR5j6y0i_I0_hLJc"
  }
}
```

Gambar 3. 35 Request Contract login

Error! Reference source not found. menunjukkan alur pemrosesan *login* pada sistem *backend*. Pemrosesan kredensial dilakukan pada *AuthenticationService*, memproses logika bisnis untuk memvalidasi kredensial dan menghasilkan token JWT pada *method* *login()*. *Authentication service* menggunakan (memiliki dependensi) pada *userRepository* dan *Jwt Service*.



Gambar 3. 36 Alur pemrosesan *login*

Method *login()* pada *AuthenticationService* melakukan pemrosesan *login* seperti pada Gambar 3. 38. *User* mengirimkan kredensial berupa *email* dan *password*, sistem kemudian mencari *user* terkait untuk mencocokkan *password* yang dikirimkan oleh *user* dengan *password* yang ada pada *email*. Jika pencocokan *password* berhasil, maka kredensial yang dikirimkan oleh *user* valid. Sistem kemudian menghasilkan token JWT dengan *custom claims* seperti pada Gambar 3. 37.

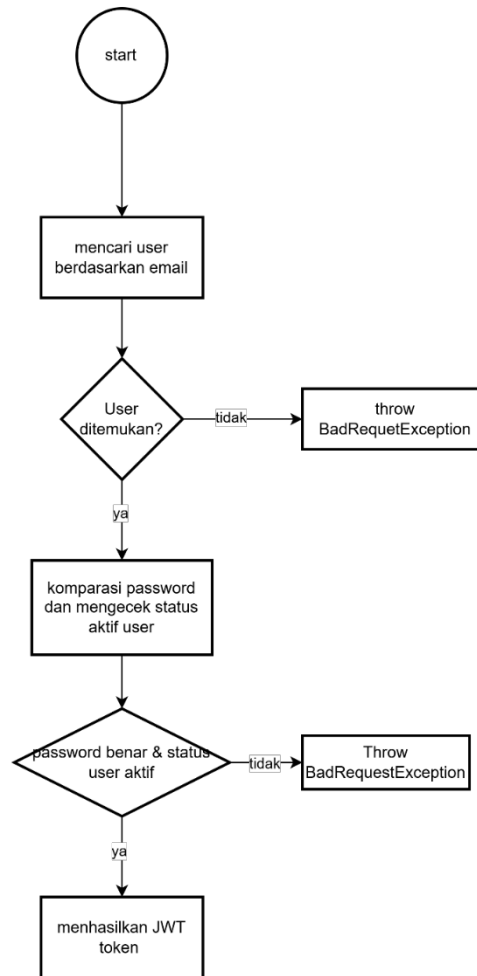
```
{
  "authorities": [
    {
```

```

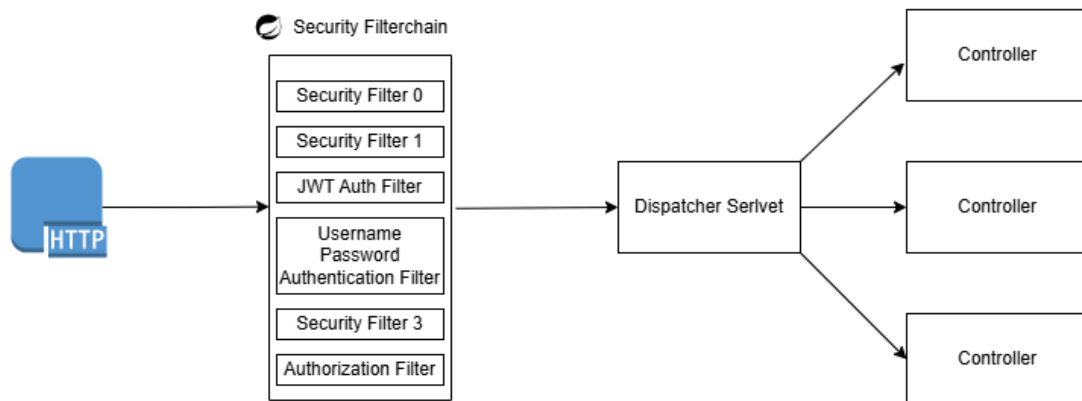
    "authority": "ROLE_USER"
  },
],
"username": "devano@gmail.com",
"isAccountNonExpired": false,
"isAccountNonLocked": true,
"isCredentialsNonExpired": true,
"isEnabled": true,
}

```

Gambar 3. 37 Claims JWT pada BSI PINTER

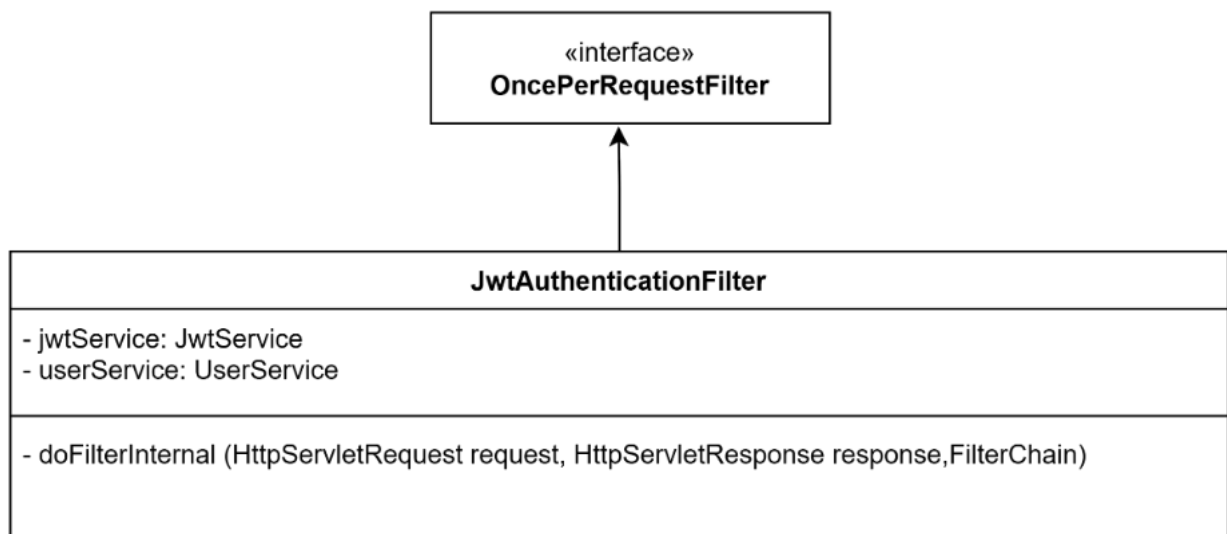
Gambar 3. 38 Proses validasi kredensial *user*

Pada Spring Security, setiap *request* yang akan diproses melalui Controller terlebih dahulu di Saring melalui Security FiterChain. Penulis menambahkan *Custom filter* yaitu JWT Auth Filter yang digunakan untuk melakukan validasi token JWT. Setiap *endpoint* yang memerlukan autentikasi maka akan disaring terlebih dahulu melalui JWT Auth Filter.

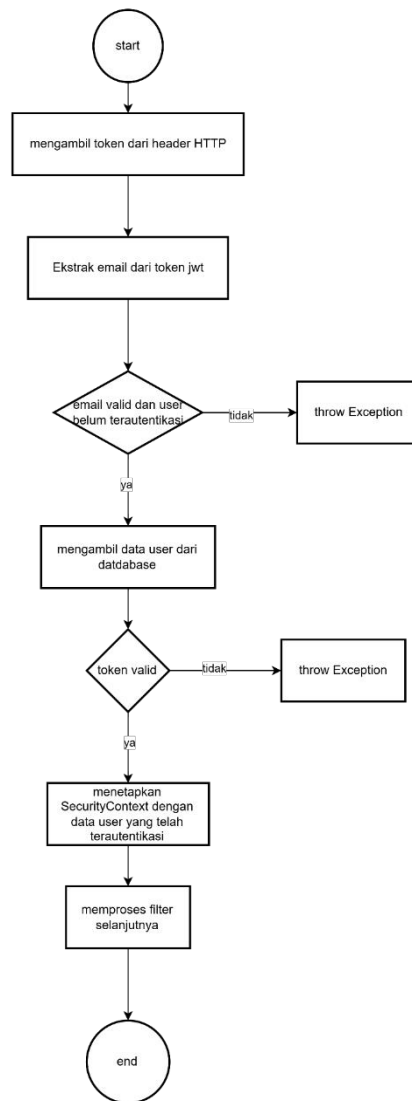


Gambar 3. 39 Alur request pada Spring Security

Berdasarkan Gambar 3. 39 token JWT disaring oleh JWT Auth Filter untuk dilakukan validasi. JWT Auth Filter merupakan sebuah Class seperti pada Gambar 3. 40 yang didaftarkan pada Security Filter Chain melalui proses konfigurasi. Class tersebut melakukan *extends* pada abstract class *OncePerRequestFilter*, melakukan *override* pada method *doFilterInternal()*. Pada sistem ini, method tersebut berisi logika yang mengeksekusi proses validasi JWT seperti pada Gambar 3. 41.



Gambar 3. 40 Class JwtAuthenticationFilter



Gambar 3. 41 Alur proses validasi JWT

Berdasarkan Gambar 3. 41, langkah awal pada proses validasi JWT adalah melakukan ekstrak *email* dari token. Sistem memverifikasi apakah *email* yang diekstrak dari token valid dan apakah pengguna belum terautentikasi. Jika *email* valid maka sistem melakukan pencarian data pengguna berdasarkan *email* pada *email*. Kemudian sistem memeriksa validasi token dengan mengecek signature JWT dan periode kadaluarsa token. Apabila token valid maka sistem akan menetapkan SecurityContext dengan informasi pengguna yang telah terautentikasi. SecurityContext memastikan bahwa informasi *user* yang telah terautentikasi dapat diakses pada *request* yang sama. Langkah terakhir, sistem kemudian mengakhiri proses filter dan melanjutkan *request* pada filter setelahnya.

Authorization

Proses otorisasi dilakukan dengan melakukan konfigurasi pada Spring security dengan menambahkan batasan akses pada *endpoint* tertentu sesuai dengan role yang dimiliki. Gambar 3. 42 menunjukkan konfigurasi yang digunakan untuk menambahkan fungsi otorisasi pada BSI PINTER. Untuk menambahkan filter otorisasi, penulis menggunakan method `hasAnyAuthority()` dengan parameter yang menyatakan hak akses pada *endpoint*.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    return http
        .csrf(AbstractHttpConfigurer::disable)
        .cors(AbstractHttpConfigurer::disable)
        .authorizeHttpRequests(auth -> {

auth.requestMatchers("api/v1/admin/**").hasAuthority("ADMIN");
        auth.requestMatchers("api/v1/dh/**").hasAnyAuthority("ADMIN",
"DH");

        auth.requestMatchers("api/v1/auth/**").permitAll();
        auth.anyRequest().authenticated();

        })
        .sessionManagement(manager ->
manager.sessionCreationPolicy(STATELESS))

        .authenticationProvider(authenticationProvider()).addFilterBefore(
            jwtAuthenticationFilter,
            UsernamePasswordAuthenticationFilter.class
        )
        .exceptionHandling(e -> {

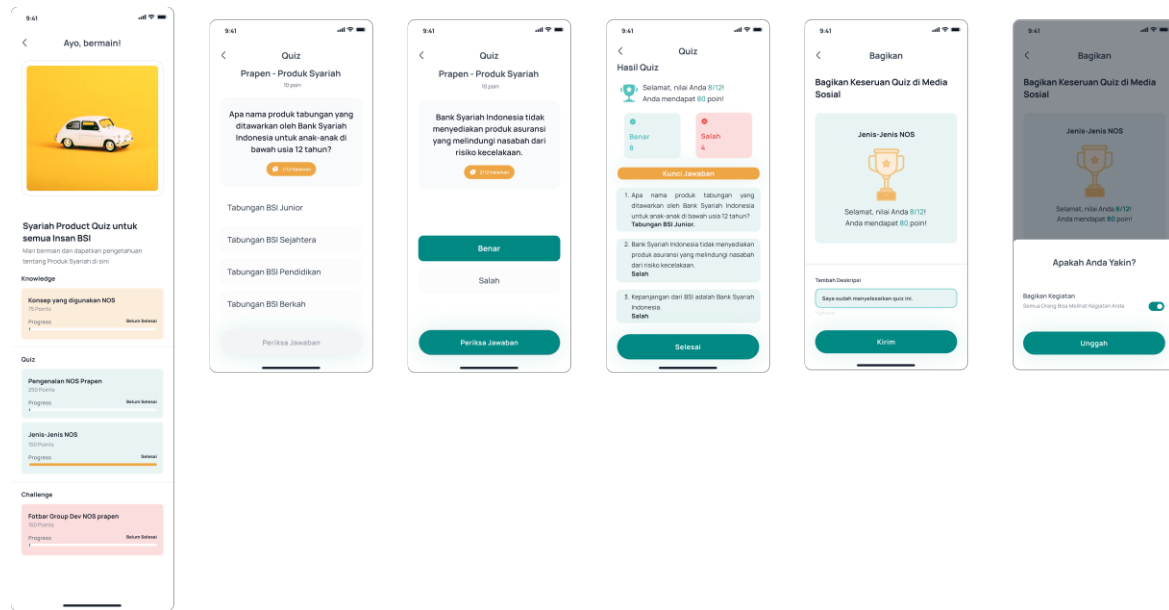
e.authenticationEntryPoint(authenticationEntryPoint());
        e.accessDeniedHandler(accessDeniedHandler());

        })
        .build();
}
```

Gambar 3. 42 Konfigurasi Spring Security

Dengan menambahkan konfigurasi maka penulis otomatis menambahkan fungsionalitas baru pada `AuthorizationFilter`. Seperti pada Gambar 3. 39 setiap *request* akan diproses melalui *filter chain*. Hasil konfigurasi otorisasi membuat `AuthorizationFilter` bertindak sesuai dengan konfigurasi yang telah ditentukan. Ketika *request* mencapai `AuthorizationFilter`, *filter* tersebut melakukan pengecekan pada hak akses *user* berdasarkan role yang dimiliki.

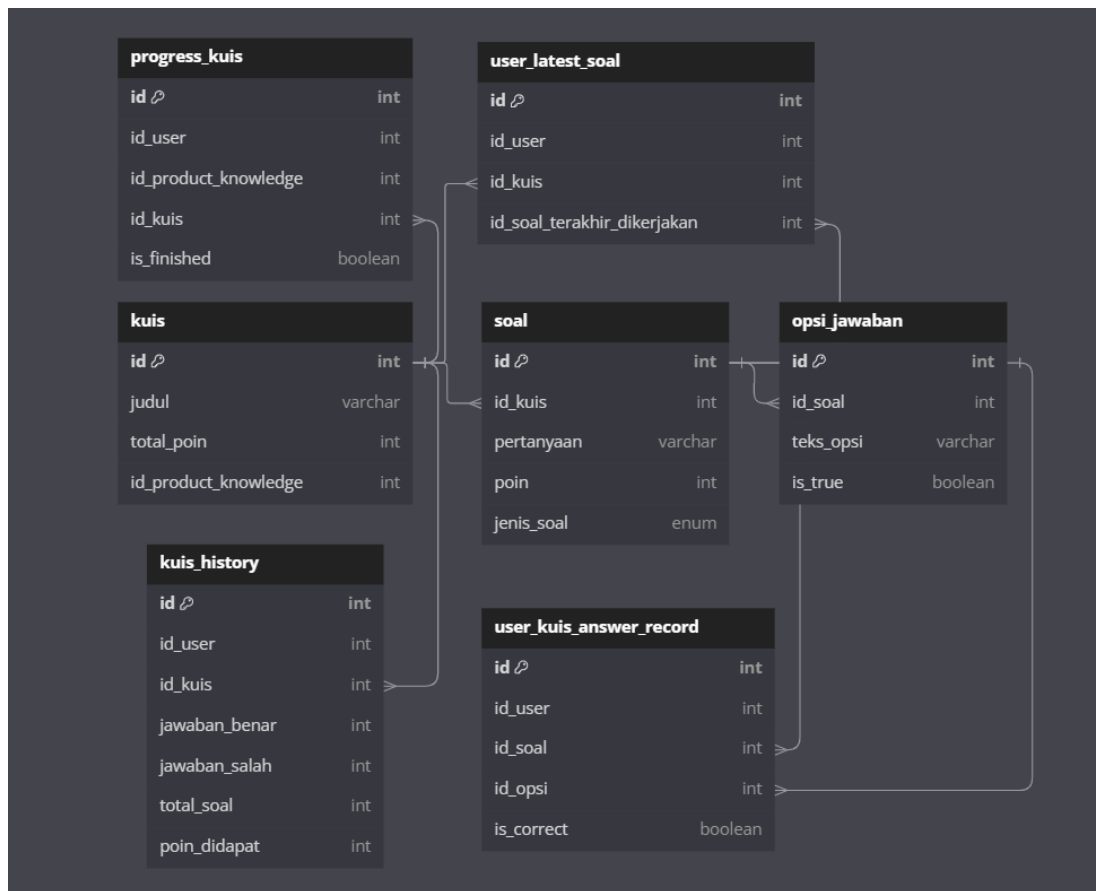
3.3.5 Implementasi Fitur Kuis



Gambar 3. 43 Alur pengerjaan kuis pada *user*

Proses pengerjaan pada fitur kuis digambarkan seperti pada Gambar 3. 43. Pertama *user* memasuki halaman product knowledge. Halaman product knowledge memiliki daftar kuis yang dapat dikerjakan oleh *user*. *User* kemudian memilih kuis yang akan dikerjakan, sistem kemudian menampilkan daftar soal yang dapat dikerjakan oleh *user*. Setelah mengerjakan seluruh soal pada kuis, *user* menyelesaikan kuis dan mendapatkan hasil pengerjaan kuis berupa poin, total jawaban benar dan salah, dan kunci jawaban. Setelah menyelesaikan kuis, *user* membagikan hasil pengerjaan kuis pada *feeds*. Dalam membagikan hasil kuis, *user* dapat memilih apakah hasil kuis akan dibagikan secara *public* atau *private*.

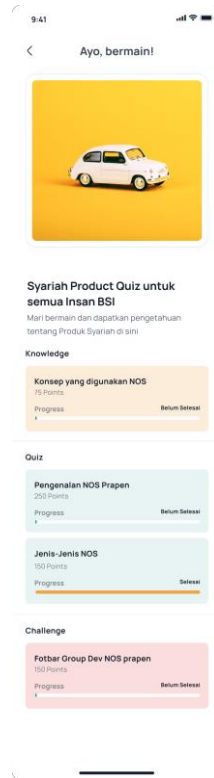
ERD



Gambar 3. 44 ERD kuis

ERD pada Gambar 3. 44 menggambarkan kebutuhan entitas untuk menjalankan fungsionalitas pada fitur kuis. Terdapat tujuh entitas yaitu *progress_kuis*, *kuis*, *kuis_history*, *user_latest_soal*, *soal*, *user_kuis_answer_record*, dan *opsi_jawaban*. Tabel *progress_kuis* digunakan untuk menyimpan progress kuis, apakah *user* sudah menyelesaikan sebuah kuis. Tabel *kuis_history* dibuat untuk menyimpan history pengerjaan kuis pada *user* untuk nantinya dikonversi menjadi hasil pengerjaan kuis. Tabel *user_kuis_answer_record* digunakan untuk menyimpan hasil jawaban benar dan salah pada opsi yang telah dijawab. Tabel *user_latest_soal* dibuat untuk menyimpan opsi terakhir yang dapat dijawab oleh *user*.

Mendapatkan daftar kuis berdasarkan product knowledge tertentu



Gambar 3. 45 Halaman *product knowledge*

Seperti pada Gambar 3. 45, terdapat bagian yang berisi daftar kuis pada *product knowledge* tertentu. *Client* perlu melakukan *request* seperti pada Gambar 3. 46, mendapatkan daftar kuis yang dapat dipilih oleh *user*. Terdapat status *isFinished* yang menandakan status *user* dalam menyelesaikan kuis. Sistem memproses *request* untuk mendapatkan *response* seperti pada Gambar 3. 47.

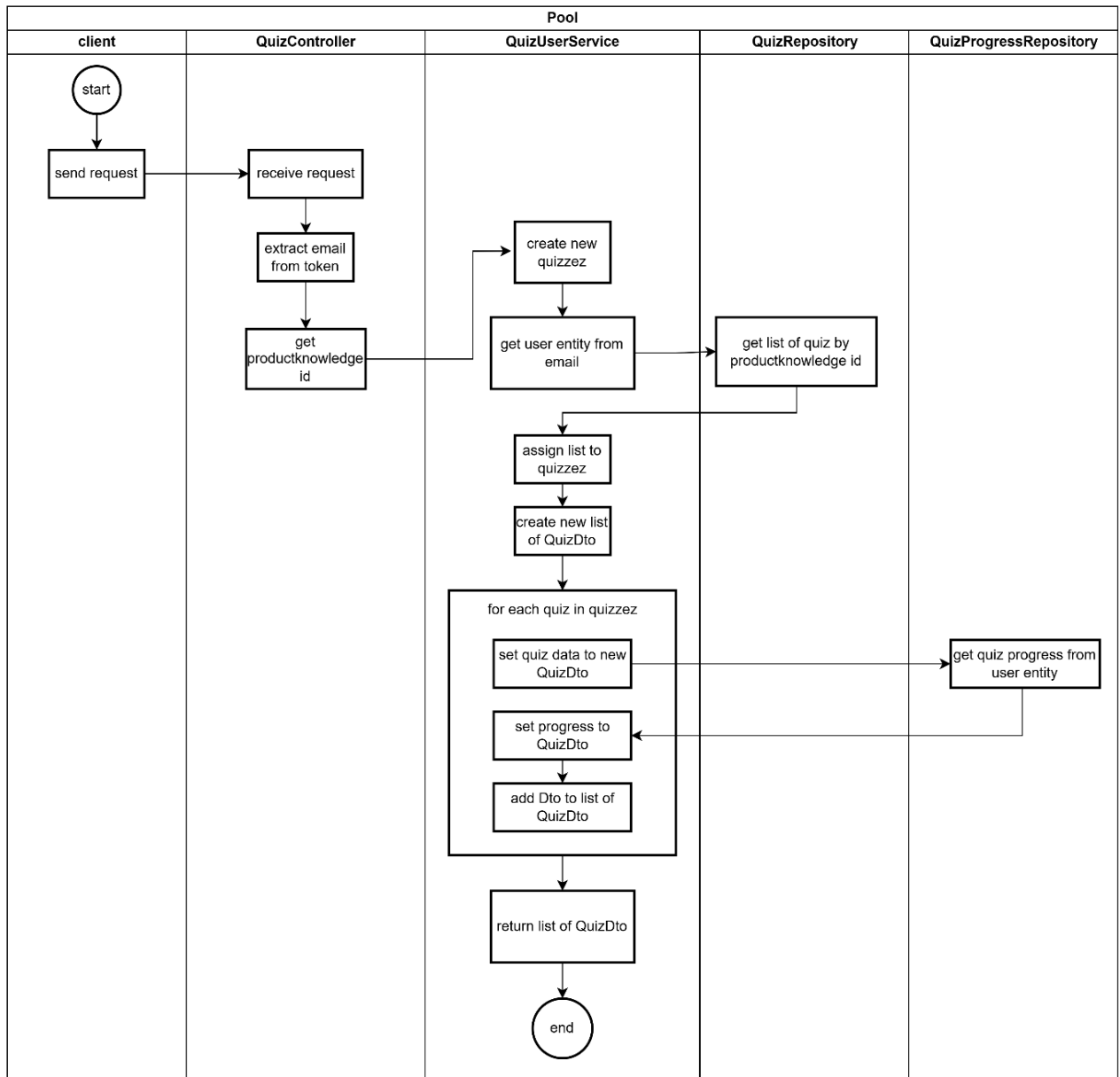
```
#GET /api/v1/quiz/productknowledge/{productknowledgeId}/quizzes
#response
{
  "status": 200,
  "message": "daftar kuis berhasil diterima",
  "data": [
    {
      "id": 1,
      "nama": "Pengenalan NOS Prapen",
      "poin": 250,
      "isFinished": false
    },
    {
      "id": 2,
      "nama": "Jenis Jenis NOS",
      "poin": 150,
      "isFinished": true
    }
  ]
}
```

```

    },
  ],
}

```

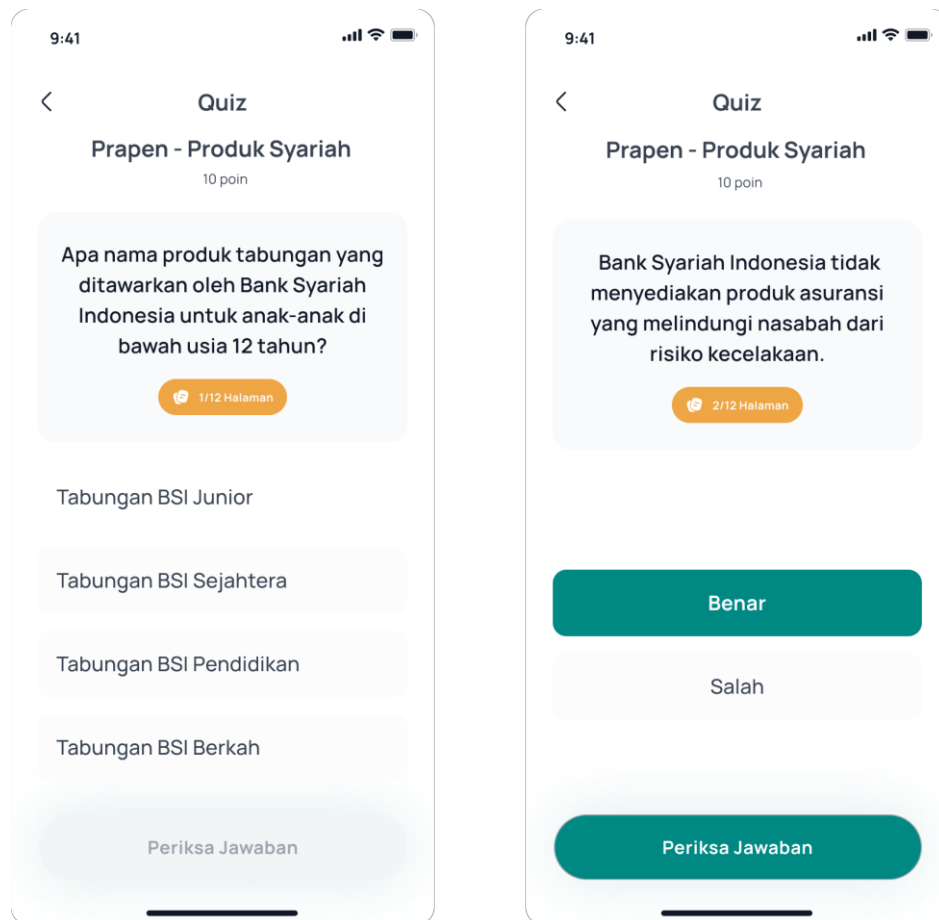
Gambar 3. 46 *Request Contract* mendapatkan daftar kuis



Gambar 3. 47 *Flowchart* mengambil daftar kuis pada product knowledge tertentu

Berdasarkan Gambar 3. 47, untuk mendapatkan daftar kuis, sistem terlebih dahulu mengambil daftar kuis dari QuizRepository. Pada setiap kuis dari daftar kuis sistem mengambil progres dari kuis dari QuizProgressRepository. Hasil akhir adalah daftar kuis yang memiliki progres pada setiap kuis nya. Daftar kuis dikembalikan pada *client* dalam bentuk JSON.

Mendapatkan detail kuis



Gambar 3. 48 Halaman detail kuis

Detail kuis merupakan daftar soal yang akan dikerjakan oleh *user*. Terdapat dua jenis soal yang ada pada kuis yaitu pilihan ganda atau benar salah. Untuk mendapatkan detail kuis, *client* melakukan *request* dan akan mendapatkan response seperti pada **Error! Reference source not found..** *Request* tersebut akan diproses oleh sistem sesuai pada Gambar 3. 50. Pada response yang dikirimkan, terdapat *idSoal* terakhir yang dikerjakan, menandakan soal terakhir yang dikerjakan oleh *user*. Berdasarkan kebutuhan, *user* dapat keluar pada saat mengerjakan soal dan kembali mengerjakan soal pada soal terakhir yang dikerjakan.

```
#endpoint
GET /api/v1/quiz/{quizId}/answer
#response
{
  "status": 200,
  "message": "Detail kuis berhasil diterima",
  "data": {
    "judul": "Prapen Produk Syariah",
    "totalHalaman": 2,
    "idSoalTerakhirDikerjakan": 0,
    "questionList": [
      {
        "id": 1,
        "poin": 10,
```

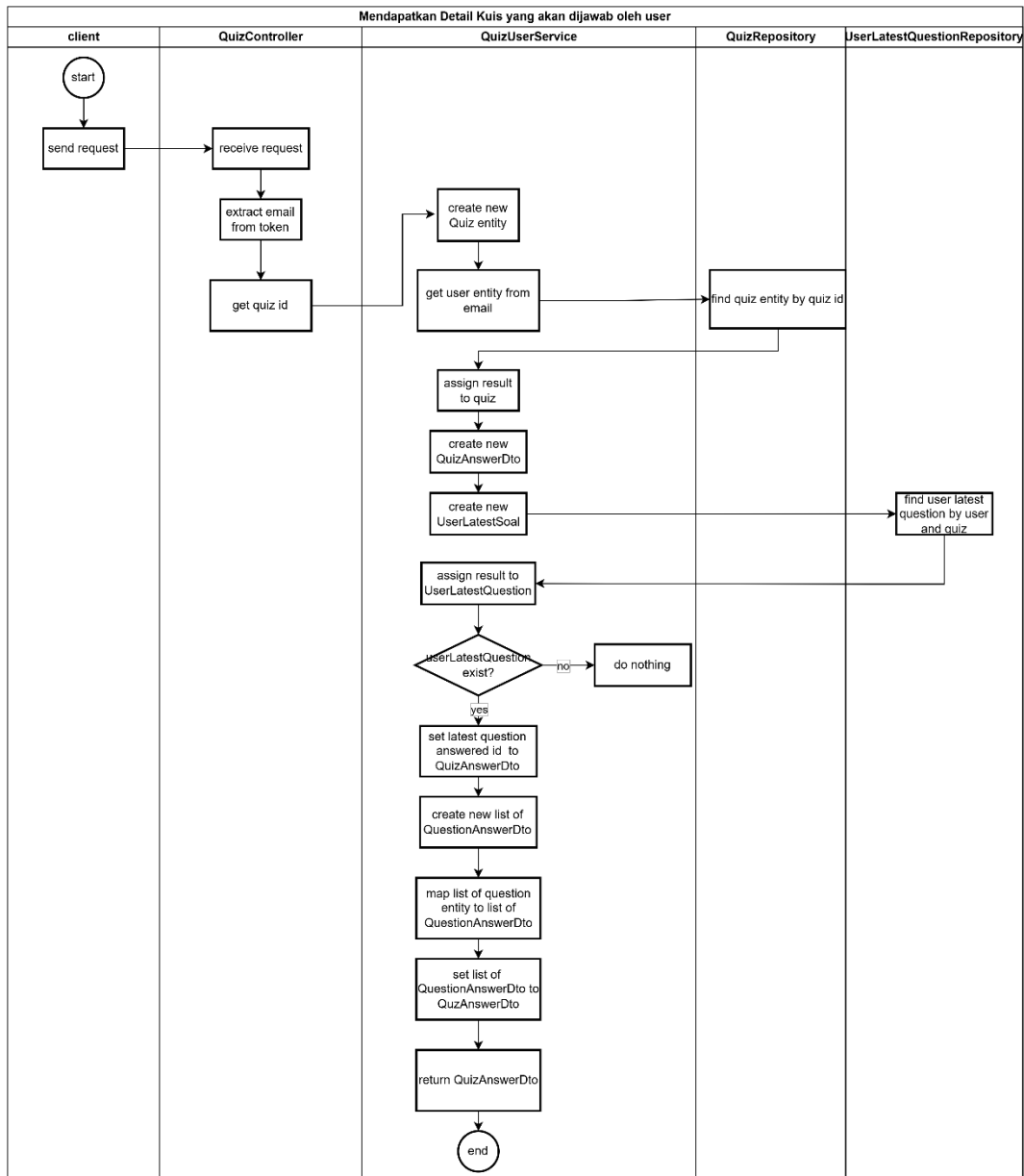
```

    "pertanyaan": "Apa nama produk terbaru tabungan yang ditawarkan Bank
Syariah Indonesia untuk anak-anak di bawah usia 12 tahun",
    "jenisSoal": "Pilihan Ganda",
    "opsiList": [
      {
        "id": 12,
        "teks": "Tabungan BSI Junior"
      },
      {
        "id": 13,
        "teks": "Tabungan BSI Sejahtera"
      },
      {
        "id": 14,
        "teks": "Tabungan BSI Berkah"
      },
      {
        "id": 15,
        "teks": "Tabungan BSI Berkah"
      }
    ]
  },
  {
    "id": 2,
    "poin": 10,
    "pertanyaan": "Bank Syariah Indonesia tidak menyediakan produk asuransi
yang melindungi nasabah dari risiko kecelakaan",
    "jenisSoal": "Benar Salah",
    "opsiList": [
      {
        "id": 16,
        "teks": "Benar"
      },
      {
        "id": 17,
        "teks": "Salah"
      }
    ]
  }
]
}
}
}

```

Gambar 3. 49 *Request* contract detail kuis

Berdasarkan *request* yang telah dikirimkan, sistem akan memproses *request* sesuai pada Gambar 3. 50. Sistem mengambil data kuis dari QuizRepository berdasarkan id kuis yang diterima. Untuk mendapatkan soal terakhir yang dijawab oleh *user*, sistem mengambil soal dari UserLatestQuestionRepository kemudian mengambil id pada soal tersebut. Hasil tersebut kemudian akan dikonversi menjadi Dto untuk kemudian dikirimkan menjadi JsonResponse.



Gambar 3. 50 Flowchart mengambil detail kuis untuk dijawab oleh insan BSI

Menjawab Soal Pada Kuis

```

#GET /api/v1/quiz/question/option/{option-id}/answer
#response
#note: jawaban benar
{
  "status": 200,
  "message": "Jawaban Anda benar",
  "data": {
    "idOpsi": 1,
    "isTrue": true,
  }
}
#note: jawaban salah
{
  "status": 200,
  "message": "Jawaban Anda salah",
  "data": {

```

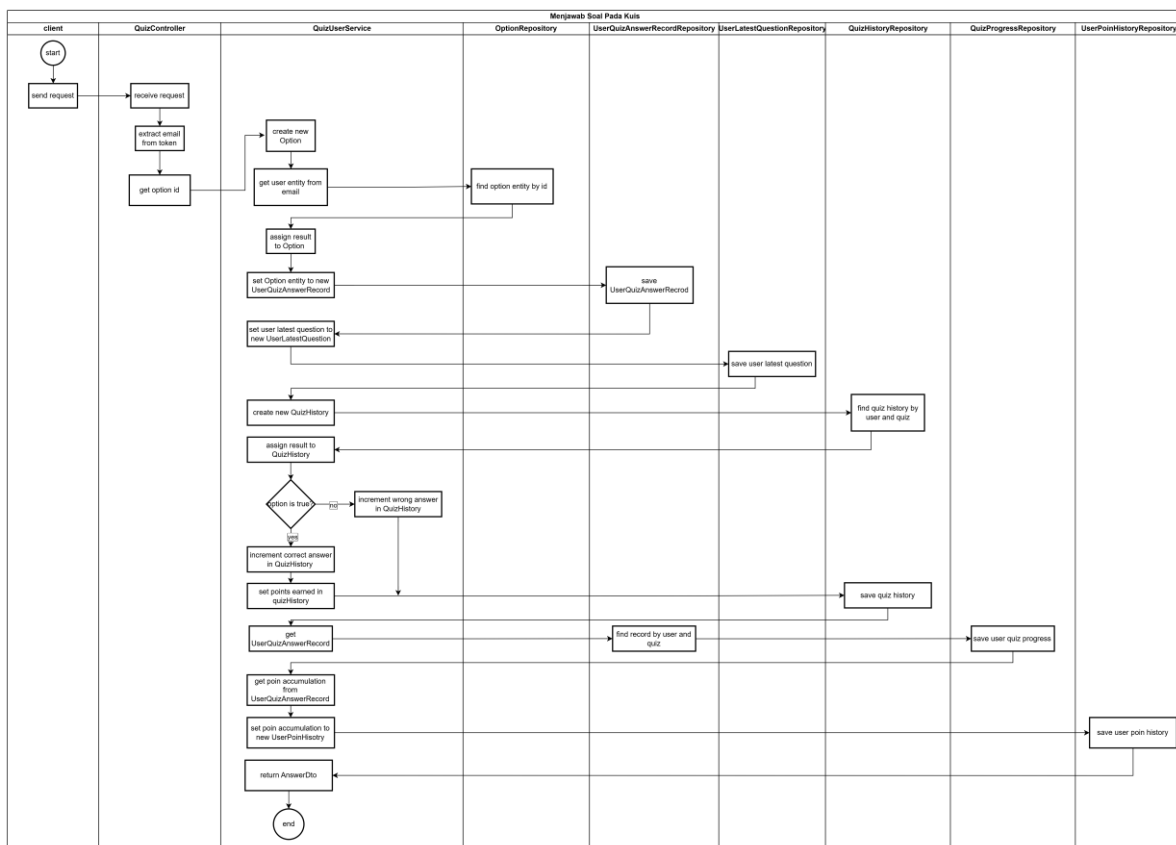
```

    "idOpsi": 2,
    "isTrue": false,
  }
}

```

Gambar 3. 51 *Request* contract menjawab soal kuis

Untuk menjawab soal pada kuis, *client* melakukan *request* dan mendapatkan response seperti pada Gambar 3. 51. *Client* mengirimkan opsi yang dipilih oleh *user* dengan mengirimkan id opsi pada query params {option-id}. Berdasarkan *request* yang dikirimkan, sistem kemudian memproses *request* sesuai *flowchart* pada Gambar 3. 52.



Gambar 3. 52 *Flowchart* menjawab soal pada kuis

Berdasarkan Gambar 3. 52 untuk memproses jawaban *user* diperlukan beberapa langkah. Pertama, sistem mengambil entitas opsi dari OptionRepository. Kemudian sistem menyimpan opsi jawaban yang dipilih pada tabel UserQuizAnswerRecord dengan UserQuizAnswerRecordRepository. Opsi jawaban kemudian disimpan pada tabel UserLatestQuestion dengan menggunakan UserLatestQuestionRepository. Sistem juga menghitung total jawaban benar dan salah kemudian menyimpan data tersebut dengan QuizHistoryRepository. Untuk menyimpan progres dari *user*, sistem menyimpan data progres pada tabel quiz_progress dengan QuizProgressRepository. Kemudian, sistem menyimpan

seluruh akumulasi poin pada *user* dengan menyimpan data menggunakan *UserPoinHistory*. Sistem memberikan respon dengan mengirimkan hasil jawaban *user* berupa benar atau salah.

Mendapatkan Hasil Akhir Kuis



Gambar 3. 53 Halaman hasil akhir kuis

Hasil akhir kuis didapatkan setelah menyelesaikan seluruh soal pada kuis. Setelah *user* menyelesaikan seluruh soal pada kuis, maka *frontend* akan menampilkan halaman seperti pada Gambar 3. 53. Untuk mendapatkan data hasil kuis pada *user*, *client* melakukan *request* dan mendapatkan response sesuai pada Gambar 3. 54. Sistem akan memproses request yang dikirimkan sesuai dengan flowchart pada Gambar 3. 55.

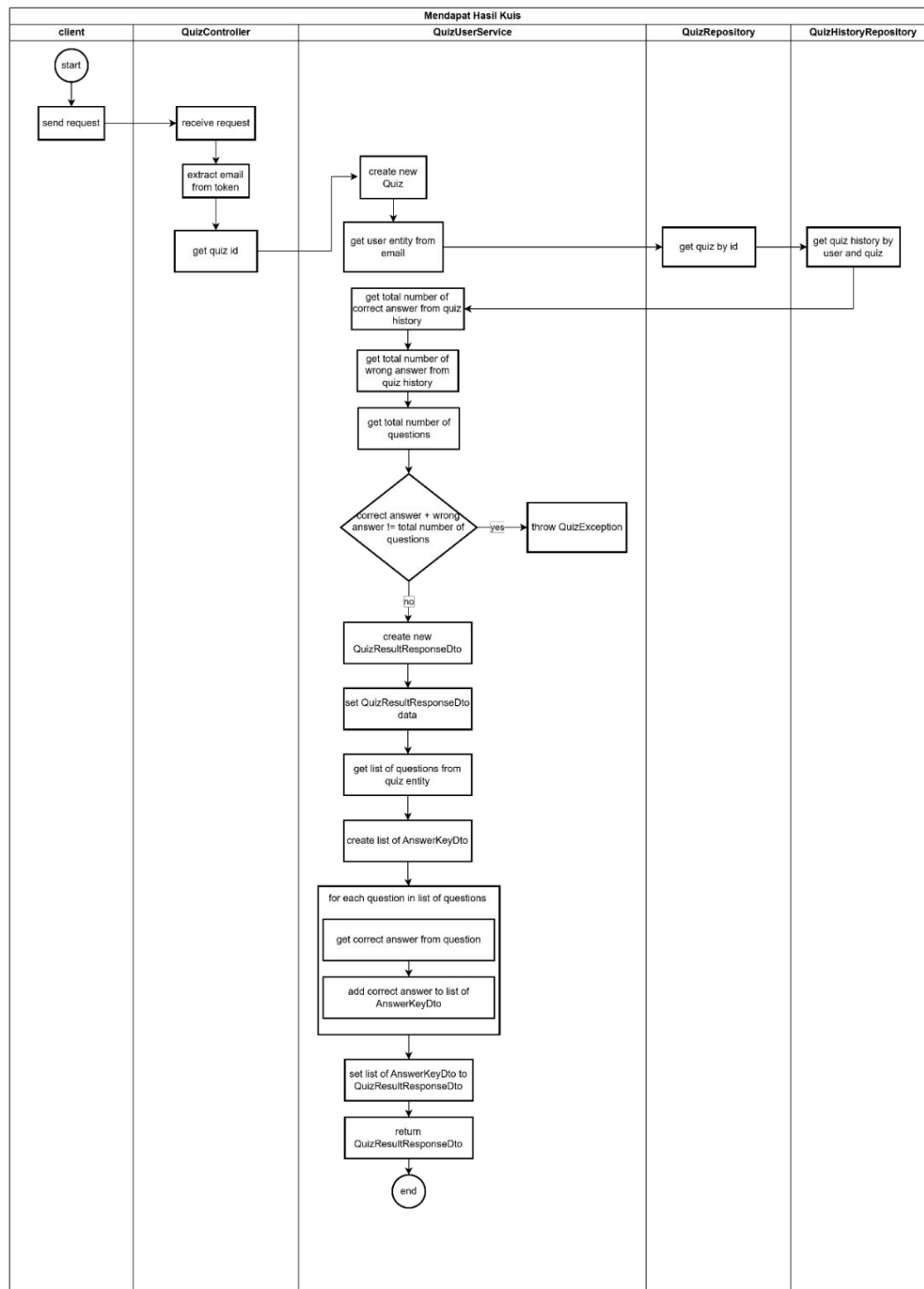
```
#GET /api/v1/quiz/{quizId}/result
#response
{
  "status": 200,
  "message": "hasil kuis berhasil diterima",
  "data": {
    "jawabanBenar": 8,
    "jawabanSalah": 4,
    "totalSoal": 12,
    "poin": 80,
    "kunciJawaban": [
      {
        "pertanyaan": "Apa nama produk tabungan yang ditawarkan oleh Bank Syariah Indonesia untuk anak-anak dibawah usia 12 tahun?",
        "jawaban": "Tabungan BSI Junior",
      },
      {
        "pertanyaan": "Bank Syariah Indonesia tidak menyediakan produk asuransi yang melindungi nasabah dari risiko kecelakaan.",
        "jawaban": "Benar",
      },
      {
        "pertanyaan": "Kepanjangan dari BSI adalah Bank Syariah Indonesia",
        "jawaban": "Salah",
      }
    ]
  }
}
```

```

    },
  ],
}
}

```

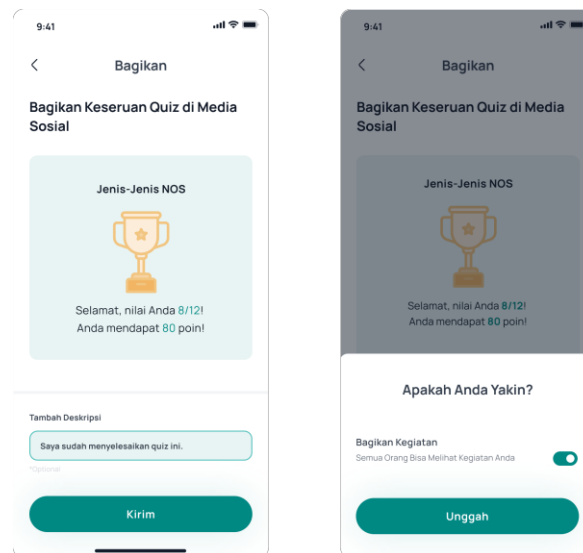
Gambar 3. 54 *Request* contract mendapatkan hasil akhir kuis



Gambar 3. 55 Flowchart memproses hasil akhir kuis

Untuk memproses hasil akhir kuis pada *user*, sistem mengeksekusi logika bisnis sesuai pada Gambar 3. 55. Sistem mengambil entitas kuis dengan mencari kuis menggunakan QuizRepository berdasarkan id kuis yang telah dikirimkan. Kemudian, sistem mengambil data hasil kuis pada tabel kuis_history dengan QuizHistoryRepository. Sistem mengembalikan respon dalam bentuk JSON, data hasil kuis yang telah dikerjakan oleh *user*.

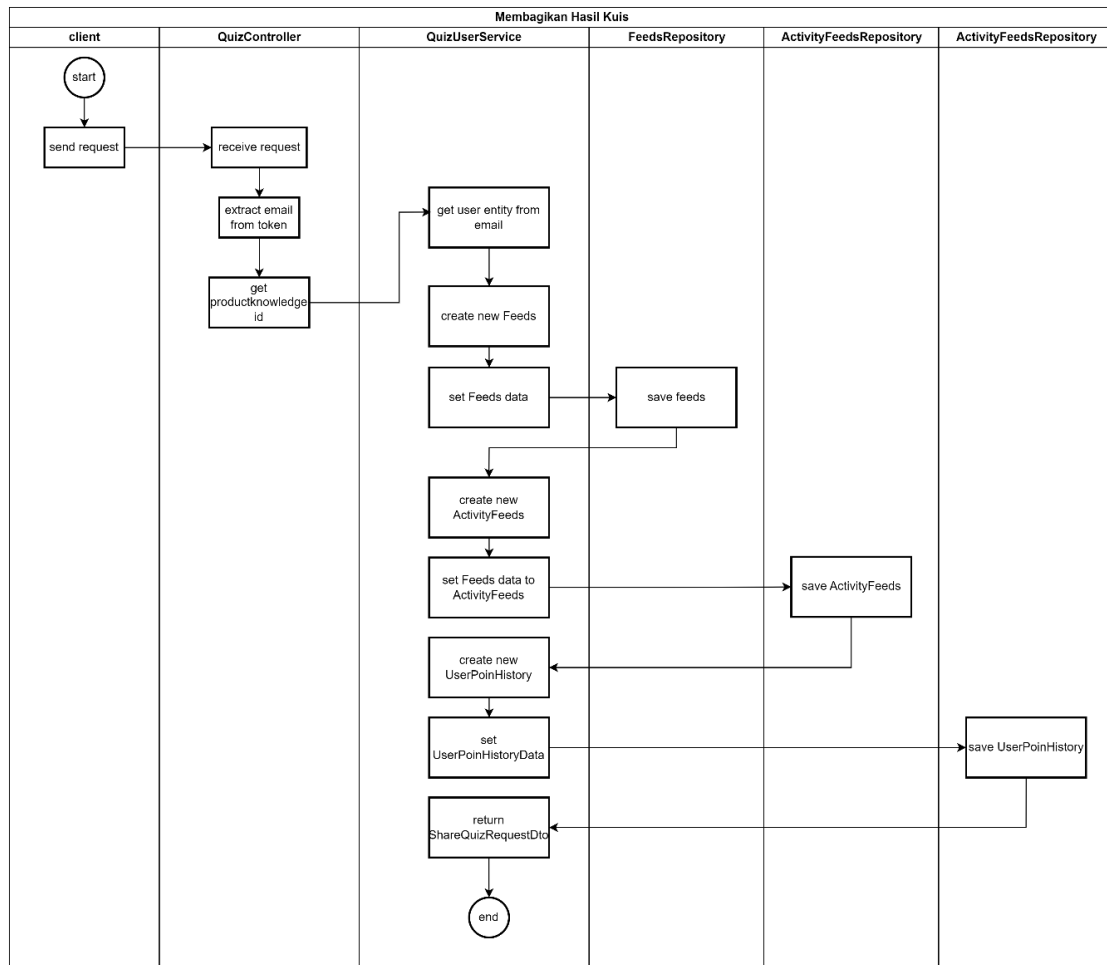
Membagikan Hasil kuis



Gambar 3. 56 Halaman membagikan hasil kuis

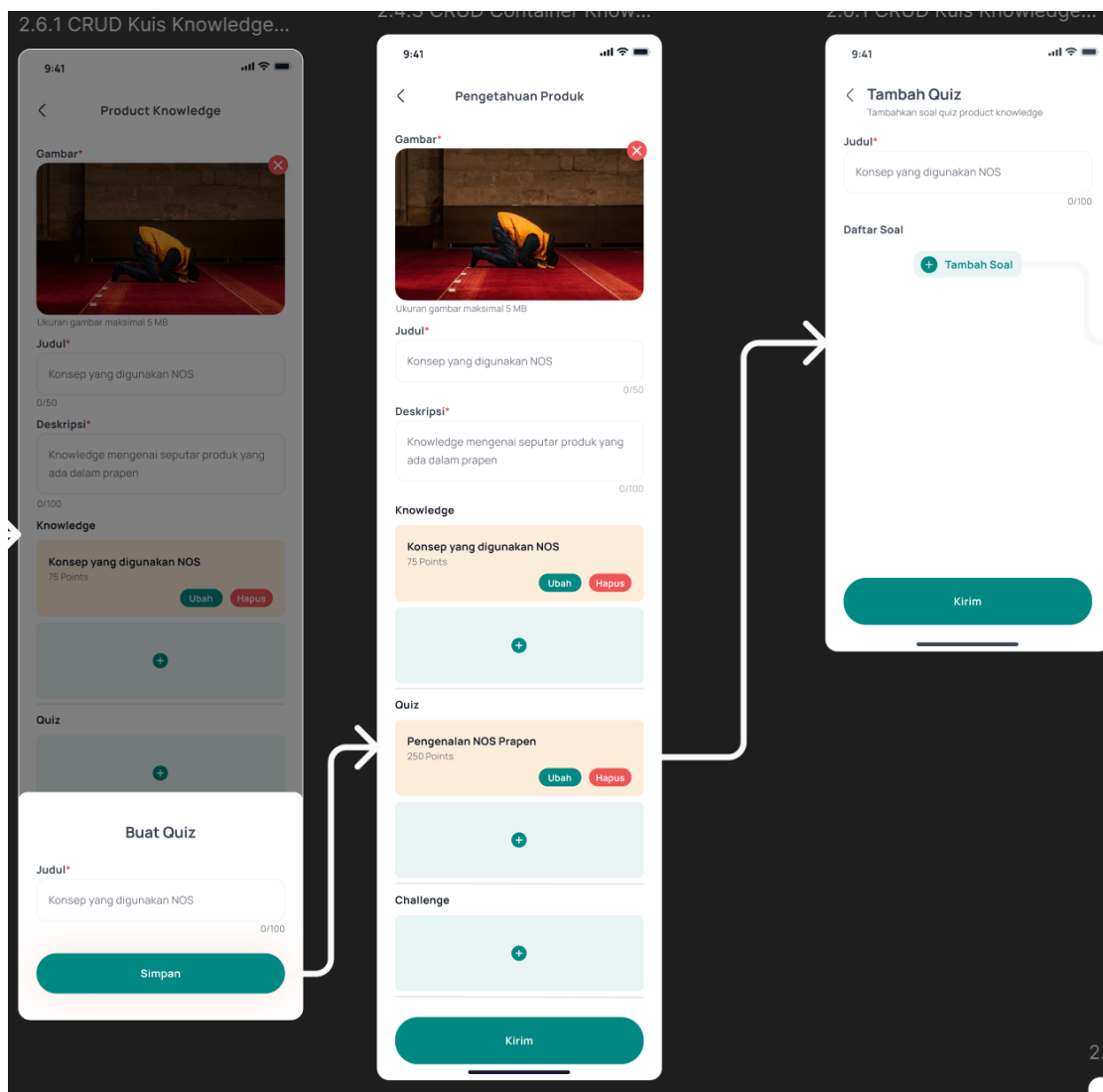
Hasil kuis dibagikan setelah *user* menyelesaikan seluruh rangkaian soal pada kuis. Setelah *user* melihat hasil kuis seperti pada Gambar 3. 53, *user* kemudian diarahkan pada halaman seperti pada Gambar 3. 56. Halaman tersebut berfungsi untuk membagikan kuis yang nantinya akan ditampilkan pada halaman *feeds*. Untuk membagikan hasil kuis, *client* melakukan *request* seperti pada Gambar 3. 57. Sistem memproses *request* sesuai pada alur seperti pada Gambar 3. 58.

```
#endpoint
POST /api/v1/quiz/share
#request
{
  "status" : 200,
  "message": kuis,
  "data":{
    "judul": "BSI NOS Prapen",
    "body": "Saya telah menyelesaikan kuis ini",
    "isShared": true,
    "poin": 40
  }
}
#response
{
  "status": 200,
  "message": "Kuis berhasil dibagikan",
  "data": null
}
```

Gambar 3. 57 *Request contract* membagikan hasil kuisGambar 3. 58 *Flowchart* membagikan hasil kuis

Untuk membagikan hasil kuis, sistem mengeksekusi logika bisnis sesuai pada Gambar 3. 58. Sistem menyimpan hasil bagi kuis dengan menyimpan sebuah entitas *feeds* dengan menggunakan *FeedsRepository*. Sistem juga menyimpan data aktivitas *feeds* dengan menyimpan entitas *activity feeds* menggunakan *ActivityFeedsRepository*. Total akumulasi poin *user* juga ditambahkan dengan menyipkan jumlah tambahan poin dengan menggunakan *UserPoinHistoryRepository*. Sistem mengembalikan response dalam bentuk JSON dengan status berhasil.

Menambahkan Kuis oleh Admin



Gambar 3. 59 Halaman menambahkan judul pada kuis

Sesuai alur pada Gambar 3. 59, untuk menambahkan kuis admin perlu menambahkan judul terlebih dahulu, menambahkan judul pada *bottom sheet* yang telah disediakan. Setelah menambahkan judul kuis, maka selanjutnya admin dapat menambahkan soal pada kuis. Soal dapat berbentuk pilihan ganda atau benar salah. Untuk menambahkan judul kuis, *client* melakukan *request* dan mendapatkan response seperti pada Gambar 3. 60. *Request* yang dikirimkan oleh *client* akan diproses oleh sistem seperti pada Gambar 3. 61.

```
#endpoint
POST /api/v1/admin/quiz/productknowledge/{id}/quiz
#request
{
  "judul":"BSI NOS Prapen"
}
#response
```

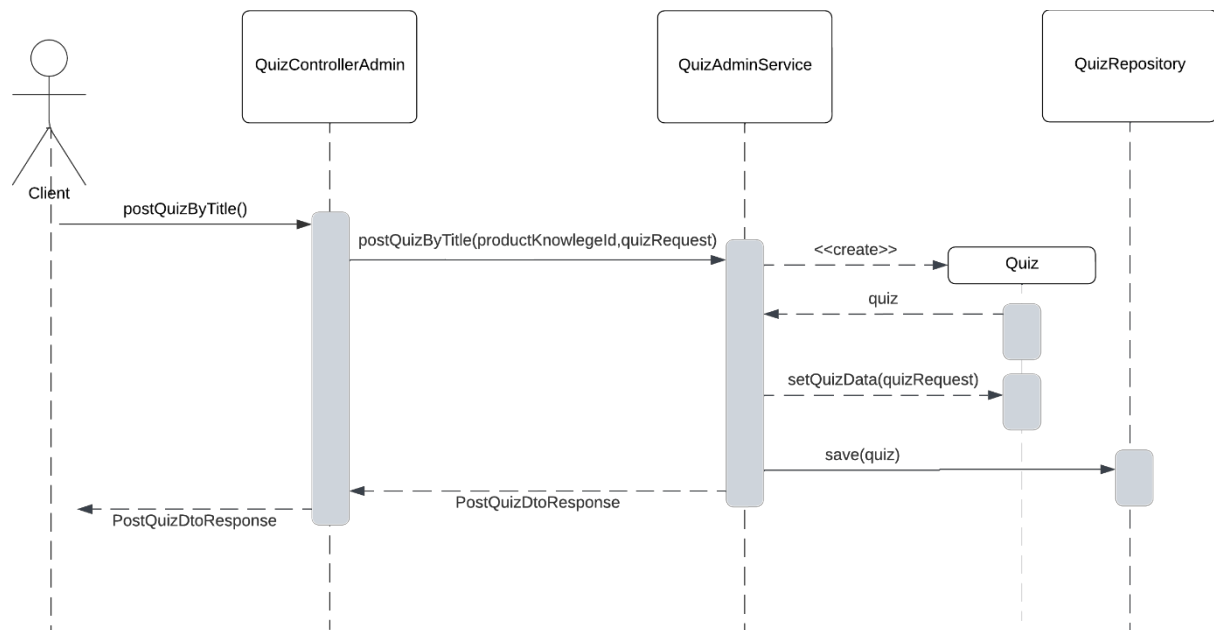
```

{
  "status": 200,
  "message": "Kuis berhasil dibagikan",
  "data": {
    "id": 1,
    "judul": "BSI NOS Prapen"
  }
}

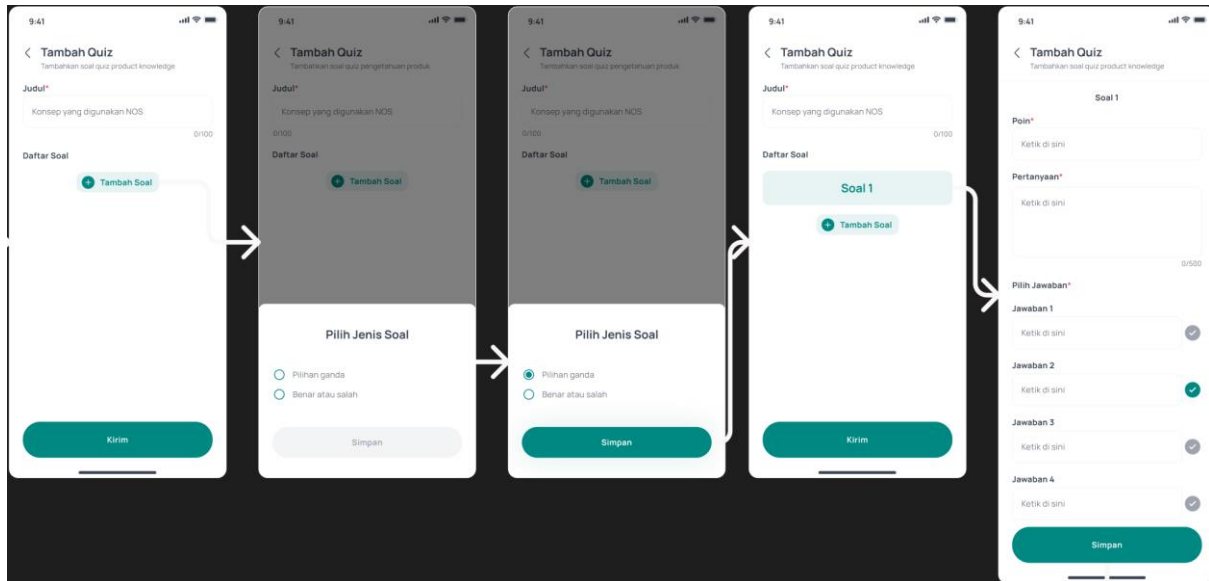
```

Gambar 3. 60 *Request* contract menambahkan judul pada kuis

Berdasarkan alur pada Gambar 3. 61, sistem memproses *request* yang dikirimkan oleh *client* dengan menambahkan entitas kuis baru. Sistem menetapkan data entitas kuis berdasarkan *request* yang dikirim oleh *client*. Sistem kemudian menyimpan entitas kuis yang telah ditetapkan dengan menggunakan QuizRepository. Terakhir, sistem mengirimkan respon pada *client* dengan mengirimkan data kuis yang telah berhasil disimpan.



Gambar 3. 61 Sequence diagram menambahkan judul pada kuis



Gambar 3. 62 Alur halaman menambahkan pertanyaan/soal pada kuis

Setelah menambahkan judul kuis, kemudian admin dapat menambahkan soal pada kuis dengan memilih tipe soal. Admin dapat menambahkan soal kuis satu per satu kemudian mengirim kuis tersebut untuk diunggah. Sebagai contoh, untuk menambahkan soal pilihan ganda *client* dapat mengirimkan *request* seperti pada Gambar 3. 63. Sistem kemudian akan memproses *request* yang dikirimkan sesuai alur pada Gambar 3. 64.

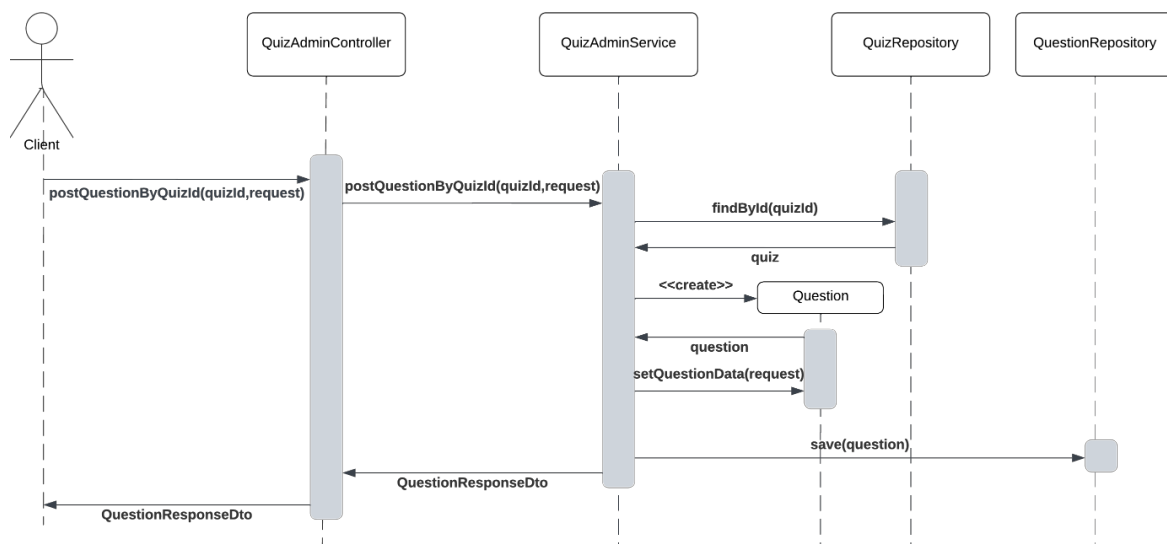
```
#endpoint
POST /api/v1/admin/quiz/{quiz-id}/question
#request
{
  "poin": 10,
  "pertanyaan": "Apa nama produk tabungan yang ditawarkan oleh Bank Syariah Indonesia untuk anak-anak di bawah usia 12 tahun",
  "jenisSoal": "pilihan_ganda",
  "opsi": [
    {
      "teksOpsi": "Tabungan BSI Junior",
      "isTrue": true
    },
    {
      "teksOpsi": "Tabungan BSI Sejahtera",
      "isTrue": false
    },
    {
      "teksOpsi": "Tabungan BSI Pendidikan",
      "isTrue": false
    },
    {
      "teksOpsi": "Tabungan BSI Berkah",
      "isTrue": false
    }
  ]
}
#response
{
  "status": 200,
  "message": "Pertanyaan berhasil ditambahkan",
  "data": {
```

```

    "poin": 10,
    "pertanyaan": "Apa nama produk tabungan yang ditawarkan oleh Bank Syariah Indonesia untuk anak-anak di bawah usia 12 tahun?",
    "jenisSoal": "pilihan_ganda",
    "opsi": [
      {
        "teksOpsi": "Tabungan BSI Junior",
        "isTrue": true
      },
      {
        "teksOpsi": "Tabungan BSI Sejahtera",
        "isTrue": false
      },
      {
        "teksOpsi": "Tabungan BSI Pendidikan",
        "isTrue": false
      },
      {
        "teksOpsi": "Tabungan BSI Berkah",
        "isTrue": false
      }
    ]
  }
}

```

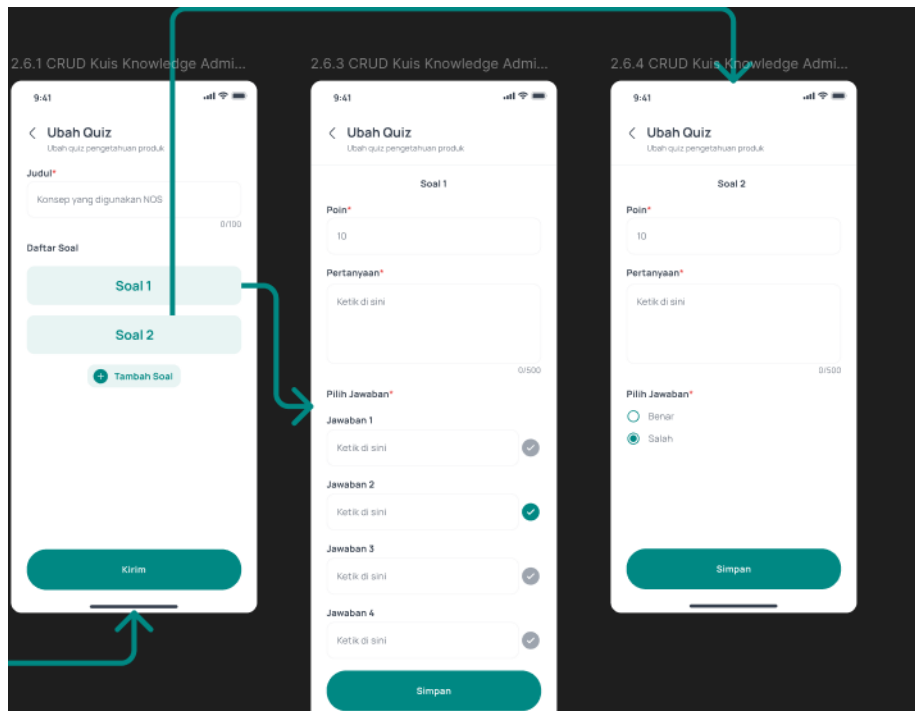
Gambar 3. 63 Request contract menambahkan pertanyaan pada kuis



Gambar 3. 64 Sequence diagram menambahkan pertanyaan pada kuis

Berdasarkan sequence diagram pada Gambar 3. 64, untuk menyimpan pertanyaan kuis pada *user* sistem menggunakan dua repository yaitu QuizRepository dan QuestionRepository. Pertama, sistem mengambil entitas Quiz berdasarkan id kuis yang dikirimkan. Sistem kemudian membuat sebuah entitas Question, kemudian menetapkan data entitas sesuai dengan *request* yang dikirimkan oleh *client*. Entitas Question kemudian disimpan menggunakan QuestionRepository. Sistem mengembalikan respon dengan mengirimkan data soal yang telah disimpan.

Memperbarui data kuis oleh admin



Gambar 3. 65 Alur halaman mengubah data kuis

Untuk memperbarui data kuis, admin dapat mengubah judul maupun mengubah soal pada kuis seperti pada Gambar 3. 65. Untuk memperbarui judul kuis, *client* mengirimkan *request* seperti pada Gambar 3. 66. *Client* mengirimkan data judul kuis yang akan diperbarui.

```
#endpoint
PUT /api/v1/admin/quiz/{id}
#request body
{
  "judul":"Pengenaln NOS Prapen"
}
#response
{
  "status":"200",
  "message":"Data kuis berhasil diubah",
  "data":{
    "id":2,
    "judul":"Pengenaln NOS Prapen"
  }
}
```

Gambar 3. 66 *Request contract* memperbarui judul kuis

Admin juga dapat memperbarui soal pada kuis. Untuk memperbarui soal pada kuis, *client* melakukan *request* seperti pada Gambar 3. 67. Admin dapat mengubah pertanyaan maupun opsi jawaban untuk memperbarui kuis. Untuk memperbarui soal pada kuis, sistem memproses *request* sesuai alur pada Gambar 3. 68.

```

#endpoint
PUT /api/v1/admin/quiz/question/{id}
#request

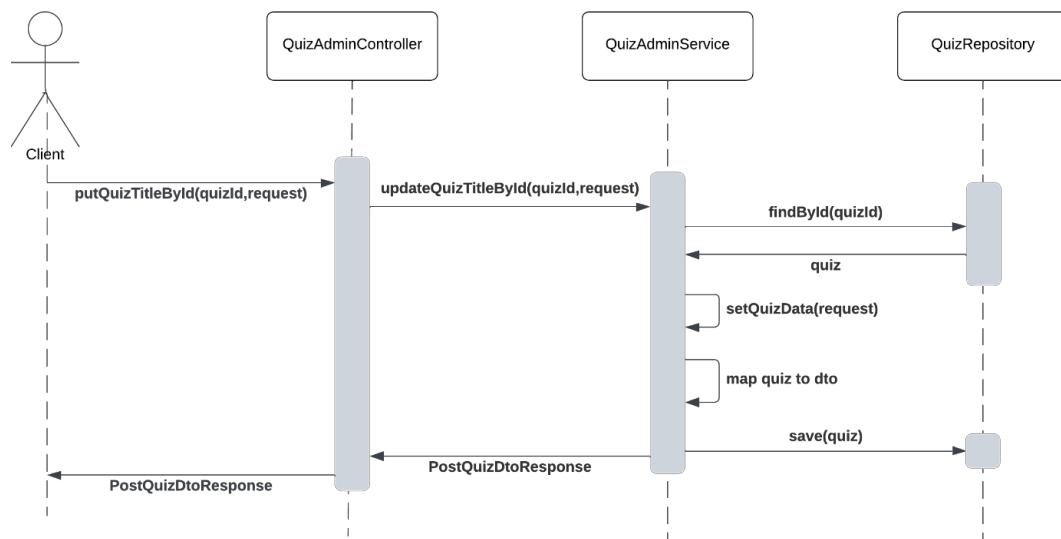
{
  "poin": 50,
  "pertanyaan": "Apa nama produk tabungan yang ditawarkan oleh Bank Syariah Indonesia untuk anak-anak di bawah usia 15 tahun",
  "jenisSoal": "pilihanGanda",
  "opsiList": [
    {
      "teksOpsi": "Tabungan BSI Junior",
      "isTrue": true
    },
    {
      "teksOpsi": "Tabungan BSI Sejahtera",
      "isTrue": false
    },
    {
      "teksOpsi": "Tabungan BSI Pendidikan",
      "isTrue": false
    },
    {
      "teksOpsi": "Tabungan BSI Berkah",
      "isTrue": false
    }
  ]
}
#response
{
  "id": 1,
  "poin": 10,
  "pertanyaan": "Apa nama produk tabungan yang ditawarkan oleh Bank Syariah Indonesia untuk anak-anak di bawah usia 15 tahun",
  "opsiList": [
    {
      "id": 1,
      "teksOpsi": "Tabungan BSI Junior",
      "isTrue": true
    },
    {
      "id": 2,
      "teksOpsi": "Tabungan BSI Sejahtera",
      "isTrue": false
    },
    {
      "id": 3,
      "teksOpsi": "Tabungan BSI Pendidikan",
      "isTrue": false
    },
    {
      "id": 4,
      "teksOpsi": "Tabungan BSI Berkah",
      "isTrue": false
    }
  ]
}

```

Gambar 3. 67 *Request* contract memperbarui pertanyaan pada kuis

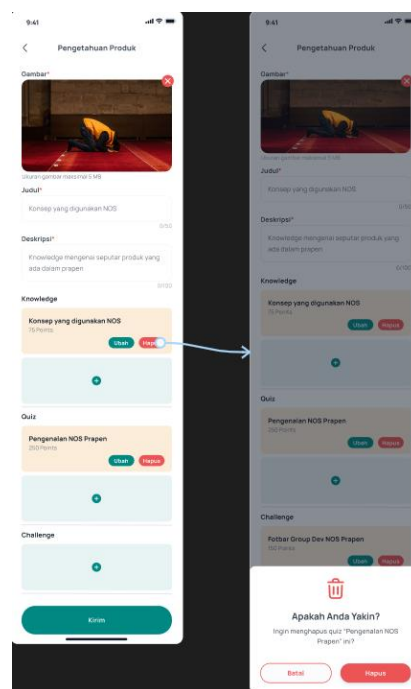
Berdasarkan Gambar 3. 68, pada QuizAdminService sistem memproses *request* dengan mengambil entitas kuis terkait berdasarkan id kuis yang dikirimkan pada *request*. Sistem menetapkan perubahan data pada entitas kuis dengan data kuis yang dikirimkan pada *request*. Kemudian entitas dengan data yang sudah diperbarui tersebut disimpan pada *email* melalui

QuizRepository. Sistem mengembalikan response berupa data kuis yang telah diperbarui dalam bentuk PostQuizDtoResponse.



Gambar 3. 68 Sequence diagram memperbarui data kuis

Menghapus Kuis oleh Admin



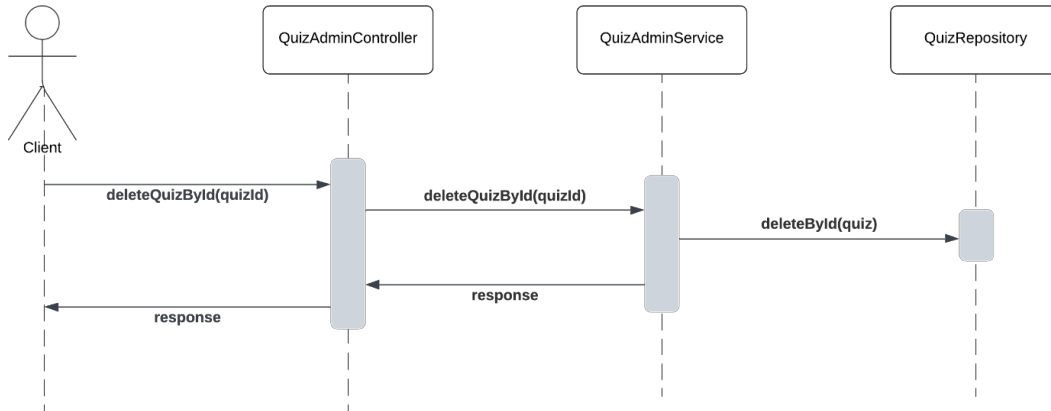
Gambar 3. 69 Alur halaman menghapus kuis

Untuk menghapus kuis, admin dapat menekan tombol hapus pada halaman *product knowledge* seperti pada Gambar 3. 69. *Client* melakukan *request* sesuai pada Gambar 3. 70 mengirimkan id kuis yang akan dihapus melalui *path variable* {id}. Sistem memproses *request* yang dikirimkan sesuai pada Gambar 3. 71.

```
#endpoint
DELETE /api/v1/admin/quiz/{id}
```

```
#response
{
  "status":200,
  "message":"Kuis berhasil dihapus",
  "data": null
}
```

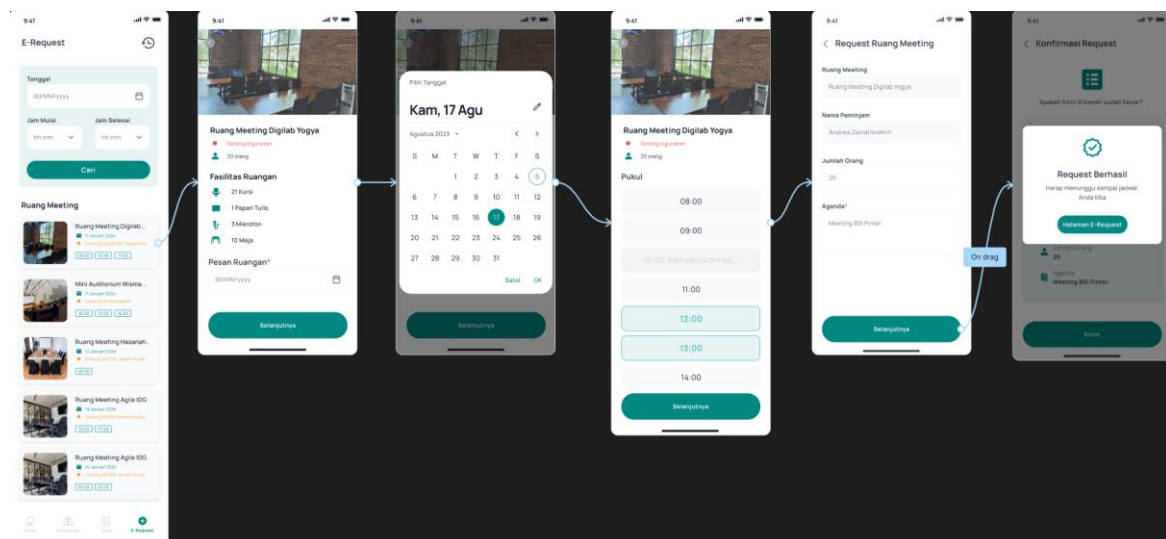
Gambar 3. 70 Request contract menghapus kuis



Gambar 3. 71 Sequence diagram menghapus kuis

Gambar 3. 71, menunjukkan alur sistem untuk menghapus kuis. Pada QuizAdminService, sistem mengeksekusi logika bisnis dengan menghapus entitas kuis melalui QuizRepository. QuizAdminService memanggil method deleteById(quizId) pada QuizRepository dengan mengirimkan argumen id kuis yang akan dihapus. Sistem mengembalikan *response* dengan status bahwa kuis telah dihapus.

3.3.6 Implementasi Fitur Pemesanan Ruang Meeting

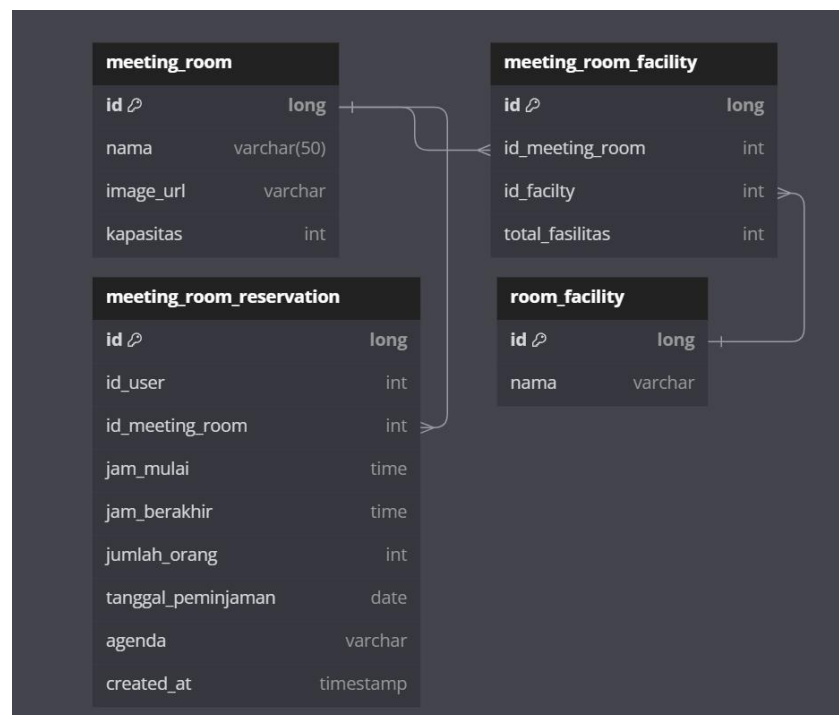


Gambar 3. 72 Alur halaman pemesanan ruang meeting

Pada fitur ruang *meeting*, halaman utama menunjukkan daftar ruang *meeting* yang dapat dipesan oleh insan BSI (*user*) seperti pada Gambar 3. 72. Untuk melakukan pemesanan ruang *meeting user* dapat menekan pada salah satu *item* ruang *meeting*, kemudian *user* akan diarahkan

pada halaman detail ruang *meeting* (nomor dua dari kiri). Kemudian *user* mengisi tanggal dan jam pemesanan. Kemudian *user* mengisi data pemesanan yaitu agenda dan jumlah orang yang akan menghadiri *meeting*. Pada kolom nama ruang *meeting* dan nama peminjam, sistem mengisi secara otomatis berdasarkan data yang telah disimpan sebelumnya. Untuk menyelesaikan pemesanan, *user* menekan tombol selanjutnya dan menekan tombol kirim pada halaman konfirmasi (halaman terakhir).

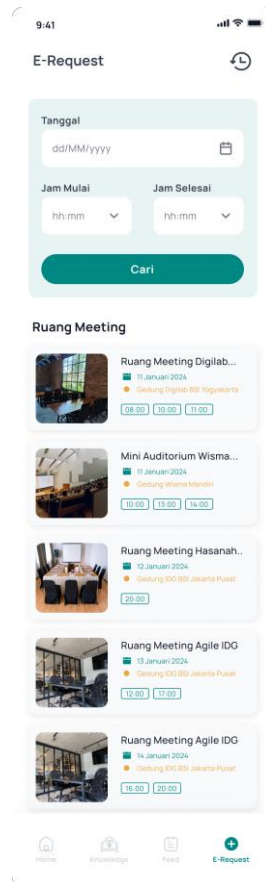
ERD



Gambar 3. 73 ERD fitur pemesanan ruang *meeting*

Berdasarkan kebutuhan dan *user story* untuk fitur ruang *meeting*, dibuatlah ERD seperti pada Gambar 3. 73. Terdapat empat entitas utama yaitu *meeting_room*, *meeting_room_reservation*, *room_facility*, dan *meeting_room_facility*. Tabel *meeting_rrom* menyimpan daftar seluruh *meeting* yang tersedia untuk dipesan. Tabel *meeting_room_reservation* menyimpan data pemesanan ruang *meeting* yang dilakukan oleh *user*. Ruang *meeting* dapat memiliki beberapa fasilitas, tabel *meeting_room_facility* dibuat untuk menyimpan daftar fasilitas pada setiap ruang *meeting*. Seluruh entitas tersebut mendukung fungsionalitas dalam fitur ruang *meeting*.

Mendapatkan Daftar Ruang Meeting



Gambar 3. 74 Halaman utama fitur pemesanan ruang *meeting*

Halaman utama pada fitur ruang *meeting* memperlihatkan daftar ruang *meeting* yang tersedia pada aplikasi seperti pada Gambar 3. 74. Pada setiap *item* daftar ruang *meeting* menampilkan ringkasan data ruang *meeting* dan ketersediaan ruangan. Untuk mendapatkan daftar ruang *meeting*, *client* melakukan *request* seperti pada Gambar 3. 75. Sistem akan memproses *request* sesuai alur pada Gambar 3. 76.

```
#endpoint
GET /api/v1/erequest/meeting/rooms
#response
[
  {
    "id": 1,
    "nama": "Ruang meeting Digilab BSI Yogyakarta",
    "imageUrl": "https://example.com/room-a.jpg",
    "isOccupied": true,
    "jamTersedia": [
      "10:00",
      "12:00"
    ],
    "kapasitas": 20
  },
  {
    "id": 2,
    "nama": "Mini Auditorium Wisma Mandiri",
    "imageUrl": "https://example.com/room-b.jpg",
```

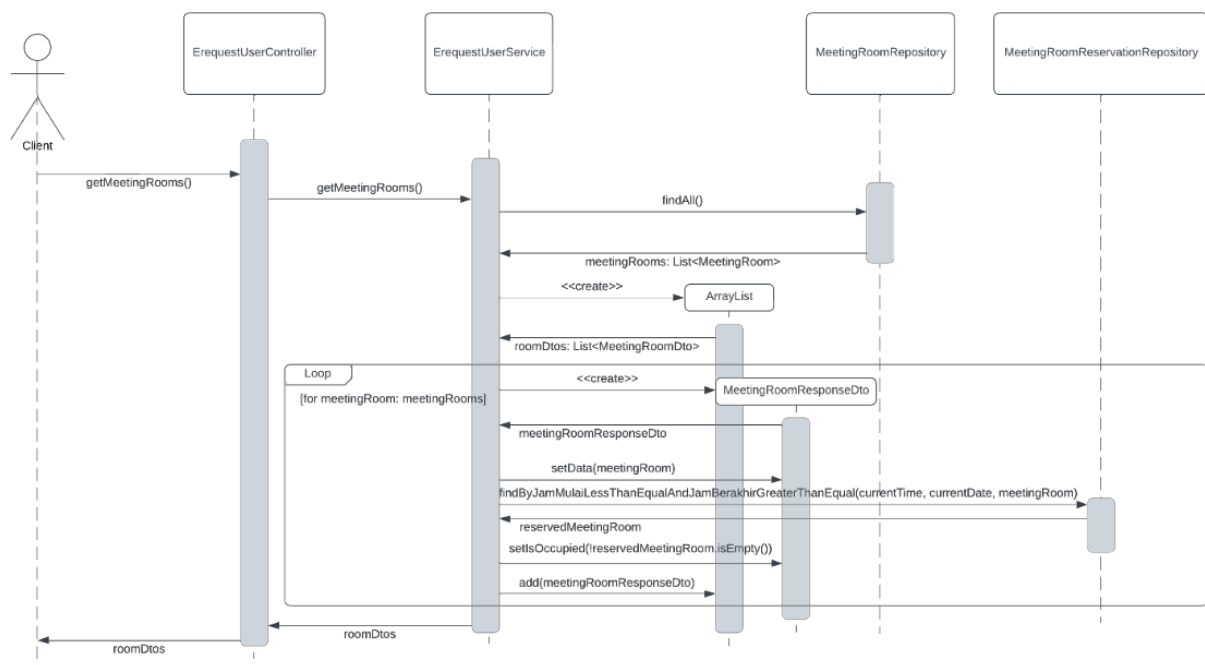
```

    "isOccupied": true,
    "jamTersedia": [
      "12:00",
      "13:00"
    ],
    "kapasitas": 30
  },
  {
    "id": 3,
    "nama": "Ruang Meeting Agile IDG",
    "imageUrl": "https://example.com/room-c.jpg",
    "isOccupied": false,
    "jamTersedia": [
      "13:00",
      "14:00"
    ],
    "kapasitas": 20
  }
]

```

Gambar 3. 75 Contract request mendapatkan daftar ruang *meeting*

Berdasarkan *request* pada Gambar 3. 75, sistem memproses *request* tersebut sesuai alur pada Gambar 3. 76. Terdapat dua repository yang digunakan, yaitu MeetingRepository dan MeetingRoomReservationRepository. ErequestService menggunakan MeetingRepository untuk mendapatkan daftar entitas ruangan *meeting*. Sistem melakukan iterasi pada setiap entitas *meeting* room kemudian menggunakan MeetingRoomReservationRepository untuk menentukan waktu ketersediaan pada setiap ruang *meeting*. Sistem mengembalikan *response* berupa daftar ruang *meeting* dan status ketersediaan pada setiap ruang *meeting*.



Gambar 3. 76 Sequence diagram mendapatkan daftar ruang *meeting*

Mendapatkan Detail Ruang Meeting



Gambar 3. 77 Halaman detail ruang *meeting*

Pada halaman utama, ketika *user* memilih salah satu *item* dalam daftar ruang *meeting*, *user* akan diarahakan pada halaman detail ruang *meeting*. Pada halaman tersebut ditampilkan daftar fasilitas dan detail data ruang *meeting*. *Client* melakukan *request* seperti pada Gambar 3. 78, mengirimkan id ruang *meeting* pada path variable {id}.

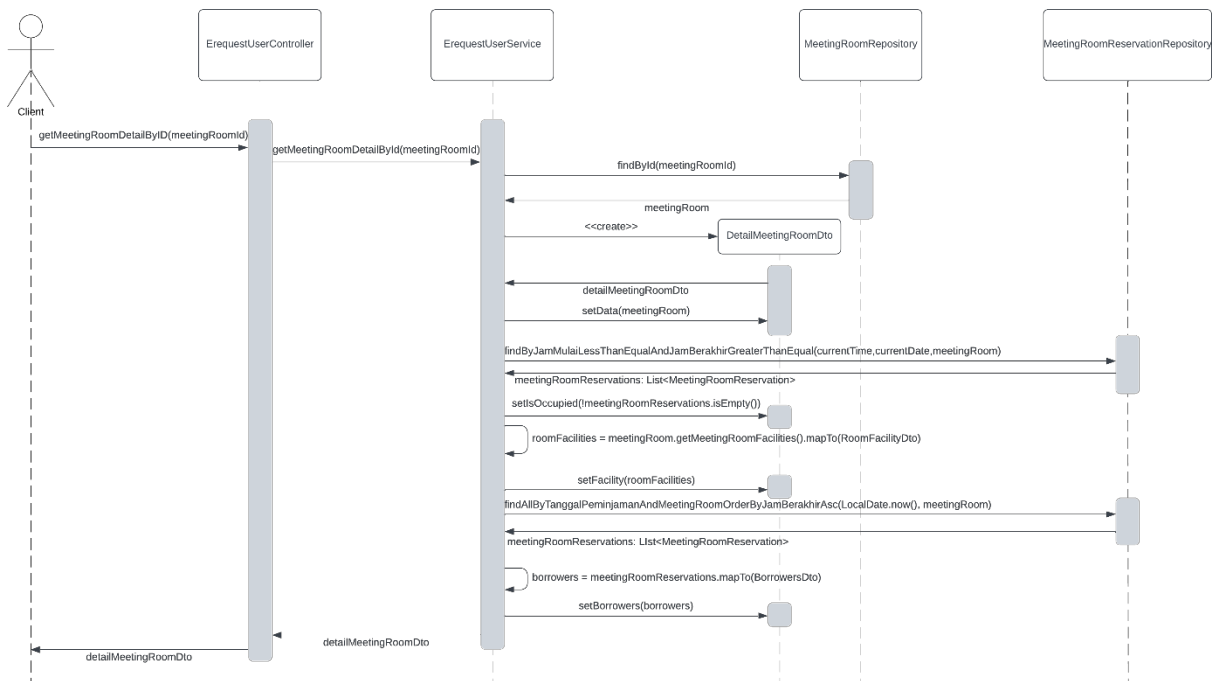
```
#endpoint
GET /api/v1/erequest/meeting/room/{id}
#response
{
  "status": 200,
  "message": "data ruang meeting berhasil diterima",
  "data": {
    "id": 1,
    "nama": "Ruang meeting digilab",
    "kapasitas": 20,
    "is_booked": false,
    "fasilitas": [
      {
        "nama": "kursi",
        "jumlah": 21
      },
      {
        "nama": "Papan tulis",
        "jumlah": 1
      },
      {
        "nama": "Mikrofon",
        "jumlah": 3
      }
    ]
  }
}
```

```

{
  "nama": "Meja",
  "jumlah": 10
},
{
  "peminjam": [
    {
      "jam_mulai": "10:00:00",
      "jam_berakhir": "12:00:00",
      "nama_peminjam": "Andrika",
      "tanggal": "2023-12-20"
    },
    {
      "jam_mulai": "15:00:00",
      "jam_berakhir": "17:00:00",
      "nama_peminjam": "Dhimas",
      "tanggal": "2023-12-20"
    }
  ]
}

```

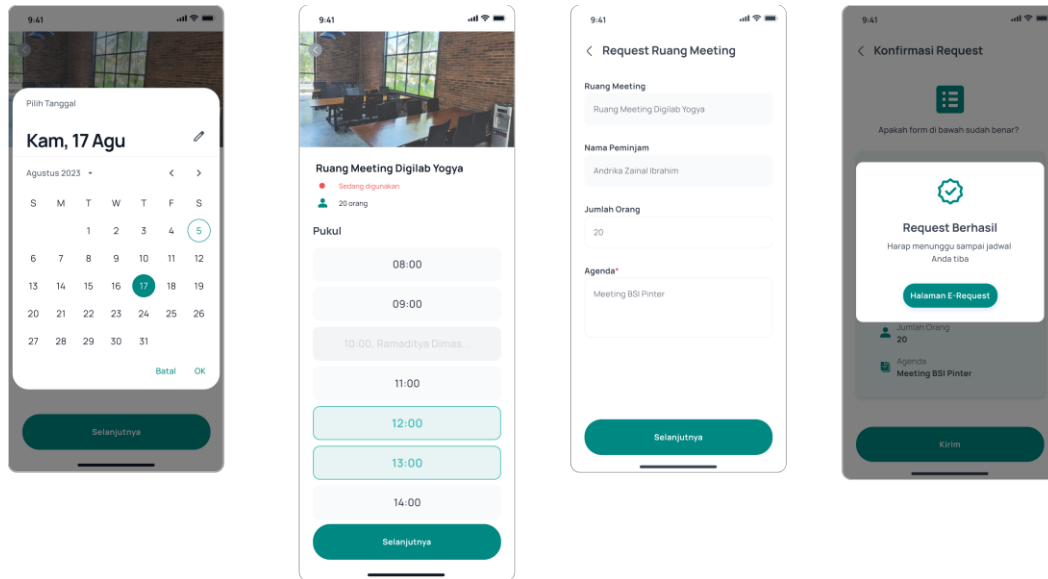
Gambar 3. 78 Request contract mendapatkan detail ruang meeting



Gambar 3. 79 Sequence diagram mendapatkan detail ruang meeting

Sistem memproses *request* yang dikirimkan oleh *client* sesuai pada Gambar 3. 79. Erequest service menjalankan logika bisnis dengan menggunakan dua komponen repository yaitu MeetingRoomRepository dan MeetingRoomReservationRepository. Sistem mengambil entitas ruang meeting melalui MeetingRoomRepository. Untuk mendapatkan daftar ketersediaan ruang *meeting* sistem menggunakan MeetingRoomReservationRepository, mengambil data daftar waktu pemesanan yang telah dilakukan oleh *user* atau *user* lainnya pada tanggal yang telah dipilih. Sistem mengembalikan respon berupa data detail ruang *meeting* beserta daftar pemesanan yang telah dilakukan oleh *user* atau *user* lainnya pada tanggal terkait.

Melakukan Pemesanan Ruang Meeting

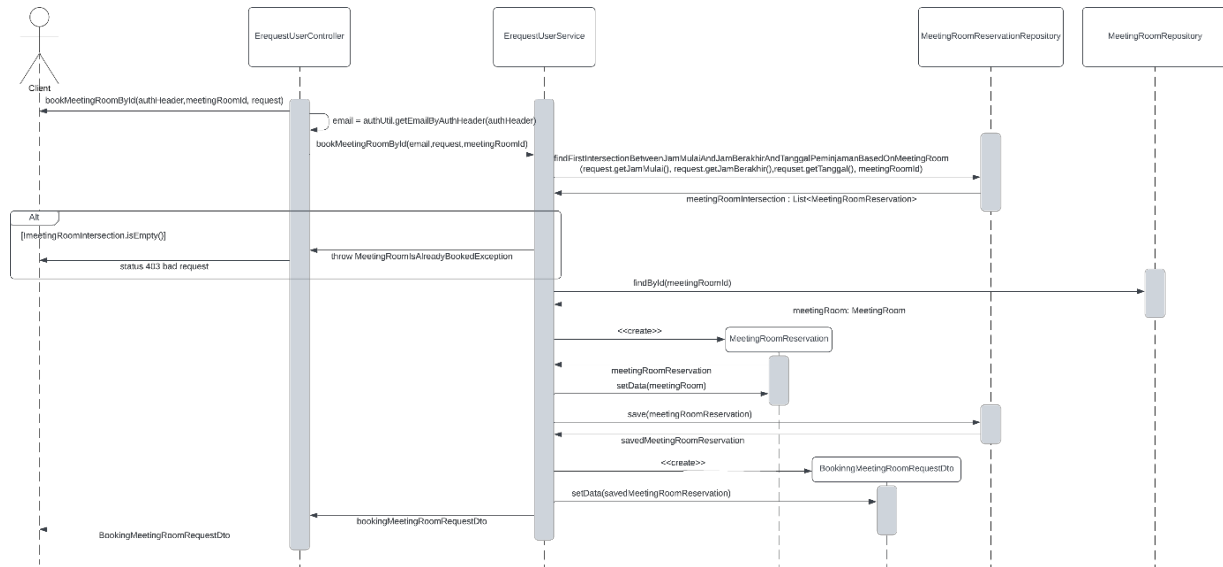


Gambar 3. 80 Alur halaman pemesanan ruang *meeting*

Setelah memasuki halaman detail ruang *meeting*, *user* dapat melakukan pemesanan ruang *meeting* sesuai alur seperti pada Gambar 3. 80. Pertama, *user* memilih tanggal pemesanan ruang *meeting*. Kemudian, *user* memilih jam mulai dan jam berakhirnya penggunaan ruang *meeting*. Terakhir, *user* mengisi detail pemesanan ruang *meeting*. *Client* melakukan *request* dengan mengirimkan *request* sesuai pada Gambar 3. 81. Sistem akan memproses *request* yang dikirimkan oleh *client* sesuai alur pada Gambar 3. 82.

```
#endpoint
POST /api/v1/erequest/meeting/room/{id}/booking
#request
{
  "tanggal": "2023-11-23",
  "jamMulai": "10:00:00",
  "jamBerakhir": "12:00:00",
  "tanggal": "12-04-2023",
  "jumlahOrang": 5,
  "agenda": "Meeting Rutin"
}
#response
{
  "status": 200,
  "message": "Ruang meeting berhasil dipesan",
  "data": {
    "tanggal": "2023-11-23",
    "jamMulai": "10:00:00",
    "jamBerakhir": "12:00:00",
    "tanggal": "12-04-2023",
    "jumlahOrang": 5,
    "agenda": "Meeting Rutin"
  }
}
```

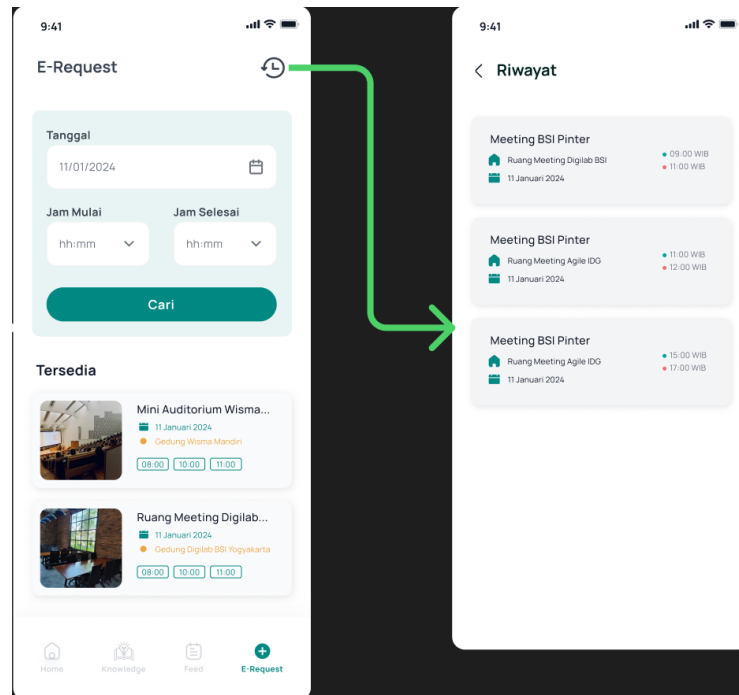
Gambar 3. 81 *Request contract* pemesanan ruang *meeting*



Gambar 3. 82 Sequence diagram pemesanan ruang meeting

Berdasarkan alur pada Gambar 3. 82, sistem menerima id ruang *meeting* yang dikirimkan oleh *client*. Untuk memproses pemesanan ruang *meeting*, pertama sistem memastikan bahwa data dikirimkan oleh *user* tidak beririsan dengan waktu pemesanan ruang *meeting* pada *user* lain. Kemudian, sistem mengambil entitas ruang *meeting* melalui *MeetingRoomRepository* berdasarkan id yang telah diterima. Sistem membuat sebuah entitas baru untuk *meeting room reservation* memasukkan data pemesanan dan mengasosiasikan entitas ruang *meeting* pada *meeting room reservation*. Sistem menyimpan data pemesanan ruang *meeting* melalui *MeetingRoomReservationRepository* dengan menyimpan entitas *meeting room reservation*. Sistem mengembalikan response dalam bentuk data pemesanan ruang *meeting* yang telah dilakukan.

Mendapatkan Riwayat Pemesanan Ruang Meeting



Gambar 3. 83 Alur halaman riwayat pemesanan ruang *meeting*

Setelah berhasil melakukan pemesanan ruang *meeting*, *user* dapat melihat riwayat pemesanan ruang *meeting* yang telah dilakukan pada halaman riwayat pemesanan seperti pada Gambar 3. 83. *Client* melakukan *request* seperti pada Gambar 3. 84, mendapatkan respon daftar riwayat pemesanan ruang *meeting* yang telah dilakukan. Sistem memproses *request* sesuai alur pada Gambar 3. 85.

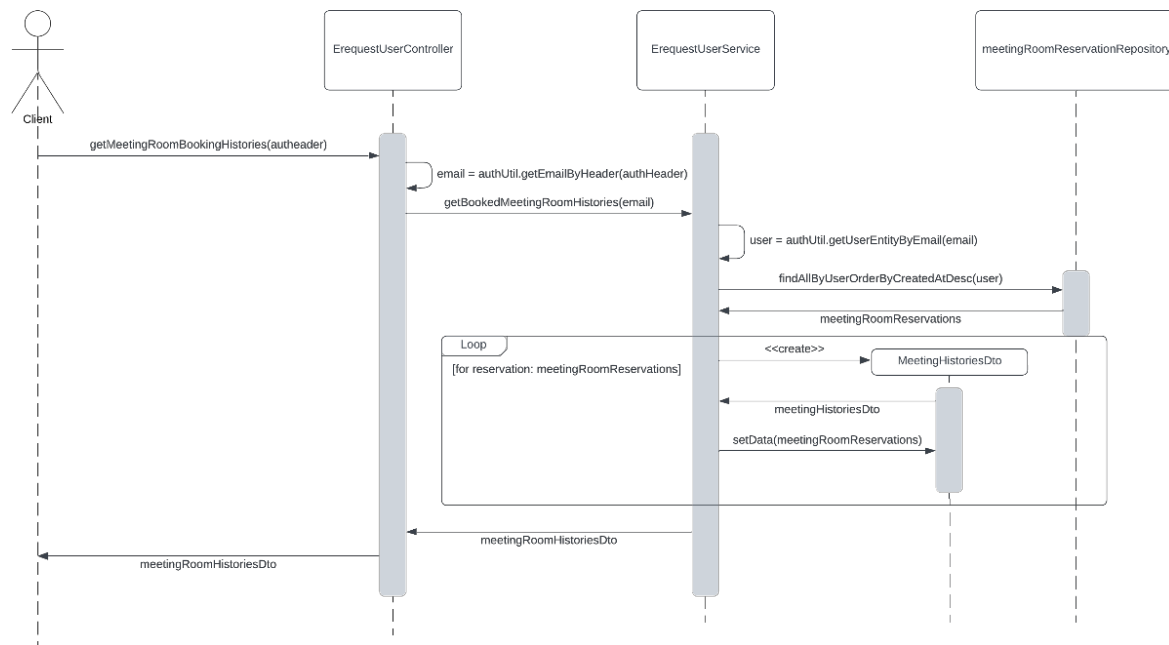
```
#endpoint
GET /api/v1/erequest/meeting/booking/history
#response
[
  {
    "id": 1,
    "agenda": "Meeting BSI PINTER",
    "namaRuang": "Ruang Meeting Digilab BSI",
    "tanggal": "2023-11-23",
    "jamMulai": "10:00:00",
    "jamBerakhir": "12:00:00"
  },
  {
    "id": 2,
    "agenda": "Meeting BSI PINTER",
    "namaRuang": "Ruang Meeting Agile IDG",
    "tanggal": "2023-11-24",
    "jamMulai": "14:00:00",
    "jamBerakhir": "16:00:00"
  },
  {
    "id": 3,
    "agenda": "Meeting BSI PINTER",
    "namaRuang": "Ruang Meeting Agile IDG",
    "tanggal": "2023-11-24",
    "jamMulai": "14:00:00",
```

```

    "jamBerakhir": "16:00:00"
  }
]

```

Gambar 3. 84 *Request contract* daftar riwayat pemesanan ruang *meeting*



Gambar 3. 85 *Sequence diagram* mendapatkan riwayat pemesanan ruang *meeting*

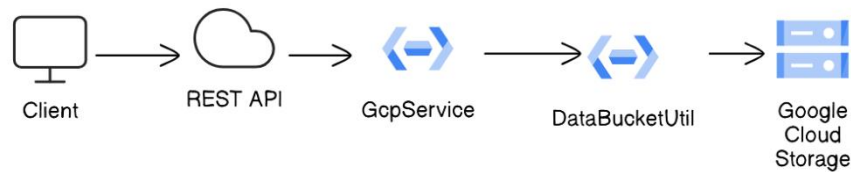
Berdasarkan alur pada Gambar 3. 85, sistem menerima *request* oleh *client* untuk mendapat riwayat pemesanan ruang *meeting*. Sesuai alur pada Gambar 3. 82, data pemesanan ruang *meeting* disimpan dalam entitas *meeting room reservation* melalui *MeetingRoomReservationRepository*. Untuk mendapatkan riwayat pemesanan ruang *meeting* pada *user*, sistem mengambil data daftar pemesanan ruang *meeting* yang telah dilakukan oleh *user* pada entitas *meeting room reservation* melalui *MeetingRoomReservationRepository*. Sistem mengembalikan respon berupa data daftar riwayat pemesanan ruang *meeting* yang telah dilakukan oleh *user*.

3.3.7 Implementasi Google Cloud Storage untuk penyimpanan fail media

Google Cloud Storage adalah layanan penyimpanan objek yang memungkinkan penyimpanan dan pengambilan data dalam bentuk objek menggunakan API Sederhana (Geewax, 2018). Penulis membuat sebuah service yaitu *GcpService* untuk mendukung fungsionalitas dalam menyimpan fail media. Dalam proyek ini, Google Cloud Storage digunakan untuk menyimpan fail media berupa gambar, setiap gambar disimpan sebagai objek dengan ekstensi fail tertentu (ekstensi yang diperbolehkan) di dalam sebuah *bucket*. *Bucket* adlaah sebuah kontainer yang berfungsi sebagai wadah untuk menyimpan dan mengelola

objek-objek tersebut secara terorganisir, memastikan akses dan kontrol yang efisien terhadap data yang disimpan.

Alur kerja



Gambar 3. 86 Alur kerja proses penyimpanan file pada BSI PINTER

Pada BSI PINTER beberapa komponen dirancang untuk mendukung proses penyimpanan *file* media secara efisien melalui integrasi dengan Google Cloud Storage. Komponen tersebut adalah GcpService, menyediakan logika bisnis untuk pengelolaan *file* media seperti melakukan proses unggah dan penghapusan *file*. Data BucketUtil merupakan *utility class* berfungsi dalam pengelolaan objek di dalam *bucket* penyimpanan.

Proses penyimpanan *file* media digambarkan sesuai alur kerja pada **Error! Reference source not found..** Proses dimulai dari *client* yang mengirimkan *request* melalui REST API pada *endpoint* yang memiliki fungsionalitas untuk menyimpan file media ke Google Cloud Storage. *Request* tersebut kemudian diteruskan ke GcpService. GcpService kemudian menggunakan DataBucketUtil untuk mengelola operasi penyimpanan objek (file media) ke dalam *bucket* di Google Cloud Storage. DataBucketUtil memastikan bahwa file media disimpan dengan nama yang unik serta diatur sesuai dengan struktur direktori yang telah ditentukan di dalam *bucket*.

Konfigurasi

Untuk mengintegrasikan Google Cloud Storage dengan aplikasi Springboot, diperlukan beberapa konfigurasi agar aplikasi dapat berkomunikasi dengan layanan Google Cloud Storage. Konfigurasi melibatkan pengaturan kredensial, pustaka, dan beberapa komponen yang diperlukan untuk mengakses *bucket* serta mengelola objek didalamnya.

Integrasi Google Cloud Platform dilakukan dengan menambahkan dependensi Spring Cloud GCP pada file pom.xml seperti pada sampel Gambar 3. 87. Pada dependensi, ditentukan versi dari dependensi ditunjukkan pada atribut <version>.

```
<dependencyManagement>
```

```

<dependencies>
  <dependency>
    <groupId>com.google.cloud</groupId>
    <artifactId>spring-cloud-gcp-dependencies</artifactId>
    <version>5.7.0</version>
    <type>pom</type>
    <scope>import</scope>
  </dependency>
</dependencies>
</dependencyManagement>

```

Gambar 3. 87 Dependensi Spring Cloud GCP pada Spring

Agar aplikasi dapat mengakses data yang tersimpan di Google Cloud Storage, aplikasi tersebut harus dilengkapi dengan kredensial yang sah. Kredensial ini berupa file JSON yang berisi informasi autentikasi, seperti ID dan kunci rahasia akun layanan. Akun layanan ini sebelumnya telah diberikan izin untuk mengakses *bucket* tertentu di Google Cloud IAM (*Identity and Access Management*). File JSON ini disimpan di lokasi yang aman pada sistem untuk melindungi kerahasiaannya. Ketika aplikasi dijalankan, *file* konfigurasi aplikasi akan berisi *path* menuju file JSON ini sehingga aplikasi dapat menemukan dan menggunakan kredensial tersebut untuk melakukan autentikasi ke Google Cloud Storage.

```

gcp.config.file=bsi-pinter-json-credentials.json
spring.cloud.gcp.project-id=bsi-pinter
spring.cloud.gcp.storage.project-id=bsi-pinter
gcp.bucket.id=bsipinter_bucket
gcp.dir.name=dev

```

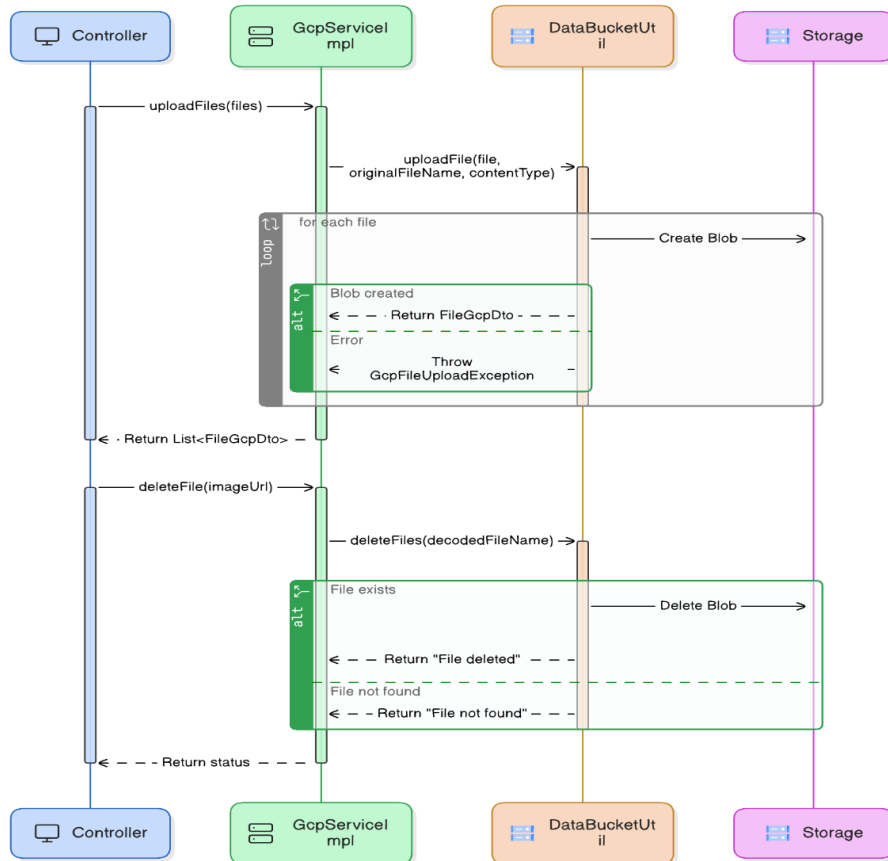
Gambar 3. 88 Contoh konfigurasi Google Cloud Platform pada BSI PINTER

Gambar 3. 88 menunjukkan contoh konfigurasi yang digunakan untuk mengintegrasikan Google Cloud Storage dengan framework Spring. Pada konfigurasi tersebut beberapa parameter ditentukan agar aplikasi dapat terhubung dengan layanan Google Cloud dan mengakses *bucket* yang tepat. Parameter "gcp.config.file" digunakan untuk menentukan lokasi *file* kredensial berisi informasi autentikasi akun layanan. *File* ini memungkinkan aplikasi untuk melakukan autentikasi terhadap Google Cloud Storage. Selanjutnya, "spring.cloud.gcp.project-id" mendefinisikan ID proyek Google Cloud yang digunakan, sedangkan spring.cloud.gcp.storage.project-id merupakan id proyek pada Google Cloud.

Parameter "gcp.bucket.id" merujuk pada nama *bucket* di Google Cloud Storage, sedangkan "gcp.dir.name" digunakan untuk mendefinisikan nama direktori penyimpanan. Seluruh konfigurasi memastikan aplikasi dapat berfungsi dengan baik dalam mengakses dan mengelola objek di Google Cloud Storage secara efisien dan aman.

Mengunggah dan menghapus fail gambar

GCP File Upload Process



Gambar 3. 89 Alur proese penambahan dan penghapusan objek di Google Cloud Storage

Proses mengunggah dan menghapus file media gambar pada Google Cloud Storage dieksekusi sesuai pada diagram Gambar 3. 89. Untuk menambahkan gambar, sistem memanggil *method* `uploadFile(files)` dengan argumen bertipe `MultipartFile`. *Files* dapat berisi satu atau lebih *file* karena bertipe array. Sistem kemudian memanggil *method* `uploadFile()` pada `DataBucketUtil` untuk membuat sebuah Blob (Binary Large Object) dari *file* yang akan diunggah pada Google Cloud Storage. Apabila *file* berhasil diunggah maka sistem akan mengembalikan `List<FileGcpDto>`. Sedangkan apabila terdapat error pada proses pengunggahan, sistem akan mengembalikan sebuah error berupa Exception.

Untuk menghapus *file* media gambar, sistem memanggil fungsi `deleteFile(imageUrl)` pada `GcpService` dengan argumen `imageUrl` yang merupakan url objek yang telah diunggah pada Google Cloud Storage. Sistem menerima url objek yang dikirimkan kemudian memisahkan url menjadi beberapa bagian untuk didapatkan nama *file* yang akan dihapus.

GcpService kemudian memanggil *method* `deleteFiles(decodedFileUrl)` dengan mengirimkan argumen berupa url objek yang telah *didecode*. Pada *method* tersebut sistem menghapus *file* berdasarkan nama *file* pada direktori yang telah didefinisikan sesuai dengan konfigurasi. Apabila *file* yang dihapus tersedia/ada, maka sistem akan mengirimkan respon bahwa *file* telah dihapus. Sedangkan apabila *file* yang akan dihapus tidak ada maka sistem akan mengembalikan eror berupa *Exception*. *Controller* akan menyesuaikan *response* status sesuai dengan kembalian yang didapat.

3.4 Hasil Pengembangan Aplikasi

Pada bab ini, disajikan hasil implementasi pengembangan dari setiap fitur yang telah dikerjakan oleh penulis. Pengembangan aplikasi ini mencakup berbagai fitur utama yang mendukung kebutuhan sistem, seperti autentikasi pengguna, registrasi, pengelolaan kuis, serta pengelolaan ruang *meeting*. Setiap fitur telah diimplementasikan dengan menggunakan REST API, yang mencakup berbagai metode dan *endpoint* untuk mendukung fungsionalitas aplikasi.

Tabel 3.5 REST URL autentikasi dan registrasi

<i>Method</i>	REST URL	Deskripsi
POST	/api/v1/auth/login	Melakukan <i>login</i> untuk autentikasi
POST	/api/v1/auth/{userId}/register	Melakukan pendaftaran akun <i>user</i> dengan mengirimkan data diri
POST	/api/v1/auth/otp-generate	Melakukan permintaan kode OTP dengan memasukkan <i>email user</i>
POST	/api/v1/auth/{userId}/otp-verify	Melakukan verifikasi kode OTP yang telah dikirimkan melalui <i>email</i>
POST	/api/v1/auth/{userId}/otp-regenerate	Melakukan permintaan ulang kode OTP

Tabel 3.5 menyajikan daftar lengkap dari seluruh *endpoint* yang dikembangkan untuk memenuhi kebutuhan autentikasi dan registrasi pengguna. Terdapat lima *endpoint* yang dirancang untuk memungkinkan pengguna melakukan *login* serta registrasi pada aplikasi. Proses registrasi melibatkan verifikasi menggunakan OTP, yang diakomodasi melalui beberapa *endpoint* khusus untuk melakukan verifikasi OTP saat registrasi. Kolom "*Method*" menunjukkan metode HTTP yang digunakan pada setiap permintaan. Kolom "*REST URL*" berisi jalur yang disertakan dalam permintaan berupa URL. Sedangkan kolom "*Deskripsi*" menjelaskan fungsionalitas dari setiap *endpoint* tersebut.

Tabel 3.6 REST URL fitur kuis

<i>Method</i>	REST URL	Deskripsi
POST	/api/v1/admin/quiz/productknowledge/{id}/quiz	Menambahkan kuis pada <i>product knowledge</i> (materi) tertentu
PUT	/api/v1/admin/quiz/{id}	Mengubah judul pada kuis
POST	/api/v1/admin/quiz/{id}/question	Menambahkan pertanyaan pada kuis
PUT	/api/v1/admin/quiz/question/{id}	Mengubah pertanyaan pada kuis
GET	/api/v1/admin/quiz/question/{id}	Mendapatkan detail daftar pertanyaan pada kuis
GET	/api/v1/admin/quiz/productknowledge/{id}/quizzes	Mendapatkan daftar kuis pada <i>productknowledge</i> tertentu
GET	/api/v1/admin/quiz/{id}	Mendapatkan detail kuis pada kuis tertentu
DELETE	/api/v1/admin/quiz/{id}	Menghapus kuis tertentu
GET	/api/v1/quiz/productknowledge/{id}/quizzes	Mendapatkan daftar kuis beserta progress pengerjaan pada <i>product knowledge</i> tertentu
GET	/api/v1/quiz/{id}/answer	Mendapatkan daftar pertanyaan pada kuis tertentu
POST	/api/v1/quiz/question/option/{id}/answer	Menjawab pertanyaan pada kuis dengan mengirimkan id jawaban
GET	/api/v1/quiz/{id}/result	Mendapatkan hasil pengerjaan kuis yang telah dikerjakan
POST	/api/v1/quiz/share	Membagikan hasil pengerjaan kuis pada <i>feeds</i>

Tabel 3.6 menyajikan daftar seluruh *endpoint* yang dikembangkan pada fitur kuis selama *sprint* satu dan dua. Terdapat tiga belas *endpoint* yang terbagi menjadi dua *role* yaitu admin dan user. *Endpoint* untuk *role* admin ditandai dengan path `api/v1/admin/**`. Kolom *method* menunjukkan *method* yang digunakan pada *request* HTTP. Kolom REST Url merupakan *path* HTTP yang disertakan pada *request* berupa URL. Sedangkan kolom deksripsi menunjukkan fungsionalitas pada URL tersebut.

Tabel 3.7 REST URL fitur pengelolaan ruang *meeting*

<i>Method</i>	REST URL	Deskripsi
GET	/api/v1/erequest/meeting/rooms	Mendapatkan daftar ruang <i>meeting</i> dan status kesediaan
GET	/api/v1/erequest/meeting/room/{id}	Mendapatkan detail data ruang <i>meeting</i>
GET	/api/v1/erequest/meeting/booking/histories	Mendapatkan histori pemesanan ruang <i>meeting</i> pada <i>user</i>
POST	/api/v1/erequest/meeting/room/{id}/booking	Melakukan pemesanan ruang <i>meeting</i>

Tabel 3.7 Menyajikan daftar seluruh *endpoint* yang dikembangkan pada fitur selama sprint tiga. Terdapat empat *endpoint* yang digunakan untuk kebutuhan *user* dalam pemesanan ruang *meeting*. Kolom *method* menunjukkan *method* yang digunakan pada *request* HTTP. Kolom Rest URL merupakan *path* HTTP yang disertakan pada *request* berupa URL. Sedangkan kolom deksripsi menunjukkan fungsionalitas pada URL tersebut.

3.5 Status Proyek

Sampai tahap penyelesaian magang, proyek BSI PINTER telah selesai dikembangkan pada fase Sprint 3. Proses eksekusi proyek terhitung dari waktu perencanaan sampai pada tahap pengembangan berlangsung selama 3 bulan. Sampai pada fase sprint 3, penulis telah mengembangkan fitur kuis, autentikasi dan registrasi serta mengembangkan fitur pengelolaan ruang *meeting* pada sisi *user*. Melakukan pengembangan dengan menyesuaikan kebutuhan yang telah ditetapkan pada *product backlog*.

Pada fase Sprint selanjutnya proyek dikembangkan sesuai kebutuhan pada sisi admin untuk fitur pengelolaan ruang *meeting*. Selain itu, pada fase pengembangan lebih lanjut ditambahkan beberapa fitur pendukung seperti integrasi dengan beberapa layanan Google Cloud untuk integrasi dengan Cloud SQL dan integrasi layanan notifikasi pada aplikasi. Proyek ini terus dikembangkan dan diuji secara berkelanjutan untuk memastikan kebutuhan dari *stakeholder* terpenuhi dan tidak ada kecacatan pada aplikasi.

Selama proses pengembangan aplikasi BSI PINTER, kebutuhan yang disampaikan oleh *stakeholder* mengalami beberapa perubahan. Sesuai dengan prinsip adaptif dari Scrum, anggota tim dituntut untuk menyesuaikan prioritas dan strategi pengembangan agar tetap selaras

BAB IV

REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Selama menjalani proses magang, penulis mendapatkan berbagai wawasan secara teori maupun praktek terkait pengembangan aplikasi. Wawasan yang didapat merupakan wawasan yang bersifat teknis maupun non teknis. Pada konteks non teknis, penulis mendapatkan wawasan mengenai penggunaan Scrum dalam pengembangan aplikasi. Penulis berkesempatan untuk terlibat secara langsung dan mengaplikasikan teori yang telah didapat di lingkungan kerja perusahaan. Pada konteks teknis, penulis mendapatkan wawasan mengenai penerapan teknologi Spring Framework dalam pengembangan aplikasi *backend*. Penulis mendapatkan kesempatan untuk berkolaborasi dengan mentor maupun anggota tim lainnya dalam mengimplementasikan teknologi Spring secara efektif dan efisien. Berdasarkan kedua konteks tersebut, penulis dapat menganalisa gap teori dan praktik yang ditemukan selama proses magang.

4.1.1 Penggunaan Spring Framework

Berdasarkan teori yang diuraikan pada Bab 2, Spring framework merupakan sebuah framework Java yang menyediakan berbagai modul untuk mempermudah aplikasi. Modul yang digunakan pada pengembangan aplikasi ini adalah Spring Boot, Spring Security, Spring Data JPA, dan Spring Cloud. Secara teoritis modul-modul ini dirancang untuk memberikan modularitas yang tinggi, kemudahan dalam manajemen dependensi, serta efisiensi dalam pengembangan sistem dengan skala besar.

Spring Core

Dalam penerapan Spring Framework, penulis merasakan adanya kemudahan yang signifikan dalam mengelola kode pada aplikasi, terutama dalam mengurangi kompleksitas kode Java yang pada umumnya dapat menjadi rumit ketika membangun aplikasi skala besar. Spring menawarkan berbagai konsep inti, seperti *Inversion of Control* (IoC) dan *Dependency Injection* (DI), yang secara teoritis bertujuan untuk memisahkan pembuatan objek dari penggunaannya, sehingga kode menjadi lebih modular, terorganisir, dan lebih mudah dikelola. Namun, dalam praktiknya, penerapan konsep DI tidak selalu sesederhana yang dijelaskan

dalam teori, dan memerlukan perhatian ekstra dalam mengelola dependensi di lingkungan pengembangan nyata.

```
@Service
@AllArgsConstructor
@Slf4j
public class KantorCabangServiceImpl implements KantorCabangService {

    private final KantorCabangRepository kantorCabangRepository;
    @Override
    public List<KantorCabang> getAllKantorCabangById(RegionArea region) {
        return kantorCabangRepository.findAllByRegion(region);
    }
}
@Repository
public interface KantorCabangRepository extends JpaRepository<KantorCabang, Long>
{

    List<KantorCabang> findAllByRegion(RegionArea idRegionArea);
}
```

Gambar 4. 1 Dependency Injection pada KantorCabangService

Gambar 4. 1 merupakan contoh dependency injection pada KantorCabangService. Pada interface KantorCabangRepository terdapat anotasi `@Repository`, dalam konsep IOC anotasi tersebut menghasilkan sebuah Bean/java object yang akan secara otomatis dikelola oleh Spring IOC. Pada KantorCabangServiceImpl, ketika menggunakan anotasi `@AllArgsConstructor` maka atribut `kantorCabangRepository` otomatis dilakukan injeksi dependency oleh Spring IOC tanpa harus mendefinisikan setter secara manual.

Secara teori, IoC dan DI memungkinkan pengembang untuk lebih mudah mengelola ketergantungan antar komponen dengan menyerahkan pengelolaan ini kepada *framework*. Namun, selama pengembangan proyek, penulis menemukan bahwa penentuan *scope* bean (*singleton*, *prototype*, dll.) dan penyusunan dependensi membutuhkan perencanaan yang cermat agar tidak menimbulkan masalah selama runtime. Misalnya, kesalahan dalam menentukan *scope* bean dapat menyebabkan inisialisasi yang tidak tepat, yang berdampak pada efisiensi program dan kinerja aplikasi, terutama ketika harus berinteraksi dengan berbagai layanan eksternal seperti Google Cloud Storage untuk penyimpanan *file* media.

Salah satu masalah yang dapat muncul dalam praktik adalah munculnya `NullPointerException` akibat proses injeksi dependensi yang tidak efektif. Hal ini bisa terjadi jika dependensi tidak berhasil diinisialisasi sebelum digunakan oleh komponen lain dalam aplikasi. Untuk mencegah hal ini, penulis harus memastikan bahwa semua dependensi diatur dengan urutan yang tepat, dan bahwa siklus hidup objek sesuai dengan kebutuhan komponen yang menggunakannya. Ini memerlukan pemahaman yang mendalam tentang manajemen siklus hidup bean serta bagaimana objek-objek tersebut dikelola oleh *container* Spring.

Selain itu, penulis juga menemukan potensi terjadinya *memory leak*, khususnya apabila secara tidak sengaja terjadi *circular dependencies*. *Circular dependencies* adalah situasi di mana dua atau lebih *bean* saling bergantung satu sama lain, sehingga menyebabkan lingkaran yang tidak bisa dipecahkan seperti yang dicontohkan pada Gambar 4. 2. Hal ini dapat menyebabkan masalah serius seperti penggunaan memori yang berlebihan atau bahkan aplikasi menjadi tidak responsif. Meskipun Spring menyediakan mekanisme untuk mendeteksi *circular dependencies*, pengembang harus tetap berhati-hati dalam menyusun arsitektur dependensi, karena dalam situasi tertentu, deteksi otomatis ini mungkin tidak selalu berhasil mengidentifikasi akar masalah.

```
@Component
public class ClassA {
    private final ClassB classB;

    public ClassA(ClassB classB) {
        this.classB = classB;
    }
}
@Component
public class ClassB {
    private final ClassA classA;

    public ClassB(ClassA classA) {
        this.classA = classA;
    }
}
```

Gambar 4. 2 Contoh *circular dependency*

Selanjutnya, kesalahan dalam konfigurasi proses injeksi dapat menimbulkan kerentanan pada tahap pengujian dan keamanan sistem. Dalam konteks pengembangan aplikasi yang berskala besar dan kompleks, pengelolaan dependensi harus dilakukan dengan cermat agar tidak hanya menjaga kinerja aplikasi tetapi juga memastikan stabilitas dan keamanannya. Salah satu pendekatan yang dipelajari oleh penulis adalah menggunakan anotasi seperti `@Autowired`, `@Component`, dan `@Configuration` dengan bijak untuk mengelola bean secara efisien, sehingga dapat mengurangi risiko kesalahan pada saat injeksi dilakukan.

Dari pengalaman ini, penulis menyimpulkan bahwa meskipun konsep IoC dan DI memberikan berbagai manfaat teoretis dalam modularitas dan efisiensi kode, penerapannya dalam proyek nyata memerlukan pemahaman yang jauh lebih dalam. Pengembang perlu berhati-hati dalam menentukan *scope* bean, menghindari *circular dependencies*, dan memastikan semua dependensi dikelola dengan benar untuk meminimalisir risiko seperti `NullPointerException` atau *memory leak*. Oleh karena itu, kontribusi akademik dari

pengalaman ini adalah pengembangan keterampilan dalam perancangan arsitektur yang lebih *robust* dan pemahaman tentang bagaimana teori dapat diterapkan secara efektif dalam kondisi pengembangan perangkat lunak yang dinamis.

Spring Security

Pada aspek keamanan, Spring Security digunakan sebagai solusi utama untuk mengamankan aplikasi berbasis Spring, menawarkan berbagai fitur keamanan yang telah teruji dan fleksibel. Berdasarkan teori, Spring Security menyediakan berbagai mekanisme untuk autentikasi dan otorisasi, yang dapat disesuaikan dengan kebutuhan sistem, termasuk dukungan untuk autentikasi berbasis token seperti JSON Web Token (JWT). Namun, dalam implementasinya, penulis menemukan adanya perbedaan signifikan antara teori yang dipelajari dan tantangan yang dihadapi selama penerapannya di lapangan, terutama ketika mengimplementasikan autentikasi berbasis JWT pada aplikasi MVC berbasis monolitik.

Secara teoretis, JWT merupakan metode yang populer untuk autentikasi, di mana token dikeluarkan setelah pengguna berhasil melakukan *login* dan disertakan dalam setiap permintaan berikutnya untuk memverifikasi identitas pengguna. Akan tetapi, dokumentasi resmi Spring Security tidak memberikan panduan yang spesifik untuk mengimplementasikan JWT pada aplikasi yang menggunakan arsitektur monolitik berbasis MVC. Hal ini mengharuskan penulis untuk mencari berbagai referensi dan praktik terbaik yang ada di komunitas untuk dapat menerapkan JWT dengan efektif.

Dalam pelaksanaan, konfigurasi autentikasi berbasis JWT memerlukan banyak penyesuaian manual agar dapat memenuhi kebutuhan spesifik perusahaan, khususnya terkait keamanan dan kontrol akses yang ketat. Salah satu tantangan utama adalah mengimplementasikan *custom filter* untuk menangani proses validasi token JWT pada setiap *request* yang masuk. Filter ini bertanggung jawab untuk memastikan bahwa setiap permintaan memiliki token yang valid dan menentukan otorisasi pengguna berdasarkan informasi yang terdapat dalam token. Proses ini jauh lebih kompleks daripada yang dijelaskan dalam teori, karena pengembang harus memastikan bahwa filter tersebut terintegrasi dengan benar dalam filter chain milik Spring Security, tanpa menyebabkan konflik dengan mekanisme autentikasi lainnya.

Selain itu, penulis juga harus memastikan bahwa implementasi JWT tidak menimbulkan celah keamanan yang berpotensi membahayakan aplikasi. Salah satu risiko utama yang harus diperhatikan adalah serangan *Replay Attack*, di mana token yang dicuri dapat digunakan kembali oleh pihak tidak berwenang. Untuk mencegah hal ini, penulis perlu menambahkan

beberapa lapisan perlindungan, seperti pengecekan keabsahan token, pengaturan masa kedaluwarsa token yang ketat, dan menerapkan mekanisme refresh token yang aman. Hal ini tidak hanya membutuhkan pengetahuan teoretis, tetapi juga keterampilan praktis dalam mengonfigurasi berbagai aspek keamanan agar sesuai dengan kebutuhan spesifik aplikasi.

Proses konfigurasi manual juga mengharuskan penulis untuk menghadapi berbagai tantangan dalam pemecahan masalah. Salah satunya adalah menangani masalah yang muncul ketika integrasi JWT tidak berfungsi sesuai harapan, yang sering kali menyebabkan permintaan pengguna ditolak atau, sebaliknya, memberikan akses yang tidak tepat. Dalam situasi ini, penulis harus memanfaatkan *debugging* secara mendalam untuk menemukan akar masalah dan memastikan bahwa setiap langkah konfigurasi, mulai dari pembuatan token hingga validasi, berjalan sesuai rencana. Implementasi *custom* filter juga memerlukan pemahaman yang lebih dalam tentang *lifecycle* filter dan bagaimana filter tersebut berinteraksi dengan berbagai komponen dalam Spring Security.

Dari pengalaman ini, penulis mendapatkan wawasan bahwa meskipun Spring Security menyediakan *framework* yang fleksibel, penerapannya di dunia nyata memerlukan keterampilan penyesuaian dan konfigurasi yang kompleks. Dokumentasi yang tersedia mungkin tidak selalu memadai untuk kebutuhan spesifik proyek, sehingga pengembang harus memiliki kemampuan riset yang baik serta pengetahuan mendalam tentang arsitektur keamanan aplikasi. Hal ini juga menggarisbawahi pentingnya pengetahuan yang memadai terkait praktik terbaik dalam mengamankan aplikasi, khususnya dalam hal penerapan token berbasis JWT, agar potensi celah keamanan dapat diminimalkan.

Secara keseluruhan, pengalaman mengimplementasikan Spring Security dengan JWT dalam proyek ini memberikan kontribusi yang signifikan dalam pengembangan kemampuan teknis penulis, terutama dalam memahami secara mendalam bagaimana teori keamanan dapat diadaptasi dan diterapkan dalam konteks yang lebih kompleks dan dinamis. Proyek ini juga memberikan wawasan berharga mengenai tantangan nyata yang mungkin dihadapi oleh pengembang perangkat lunak dalam menjaga keamanan aplikasi yang mereka kembangkan.

Spring data JPA

Spring Data JPA merupakan salah satu modul utama yang digunakan dalam pengembangan aplikasi *backend* BSI PINTER. Berdasarkan teori yang diuraikan pada Bab 2, Spring Data JPA adalah modul yang mempermudah akses data dalam aplikasi berbasis Java dengan mengurangi *boilerplate code* yang sering diperlukan untuk pengelolaan *email*. Modul

ini menawarkan mekanisme otomatis dalam pengelolaan entitas, termasuk dalam operasi CRUD (Create, Read, Update, Delete) yang umumnya memakan waktu dalam pengembangan sistem besar. Namun, dalam praktiknya, penulis menemukan beberapa perbedaan antara teori dan implementasi di lapangan.

Secara teoretis, Spring Data JPA memungkinkan pengembang hanya perlu mendefinisikan *interface* yang mengimplementasikan `JpaRepository`, sehingga proses CRUD dapat dijalankan secara otomatis tanpa perlu menulis kode SQL manual. Namun, selama proses magang, penulis menemukan bahwa penggunaan Spring Data JPA tidak selalu semudah teori yang dijelaskan. Salah satu tantangan utama yang dihadapi adalah ketika harus membuat *custom query* untuk memenuhi kebutuhan bisnis spesifik yang tidak dapat diakomodasi oleh metode bawaan (`save()`, `findAll()`, dll.) dari `JpaRepository`. Dalam hal ini, penulis harus menggunakan JPQL (Java Persistence Query Language) atau Native Query seperti pada Gambar 4. 3 untuk menyelesaikan berbagai skenario yang lebih kompleks, seperti pencarian berdasarkan kriteria yang rumit atau penggabungan data dari beberapa tabel.

```
@Query("SELECT m from MeetingRoomReservation m WHERE (m.jamMulai<:current AND
:current<m.jamBerakhir) AND m.tanggalPeminjaman=:now AND
m.meetingRoom=:meetingRoom")
List<MeetingRoomReservation>
findByJamMulaiLessThanEqualAndJamBerakhirGreaterThanOrEqualTo(LocalTime current,
LocalDate now, MeetingRoom meetingRoom);
```

Gambar 4. 3 Kueri JPA menggunakan JPQL

Selain itu, pengelolaan relasi antar entitas juga menjadi tantangan tersendiri. Secara teori, Spring Data JPA mendukung relasi antar entitas, seperti `@OneToMany`, `@ManyToOne`, dan sebagainya. Namun, dalam praktiknya, penulis menemukan bahwa pengelolaan relasi yang tidak teliti dapat menimbulkan masalah seperti `LazyInitializationException`, yang terjadi ketika aplikasi mencoba mengakses entitas terkait yang belum diinisialisasi. Hal ini memerlukan pengetahuan lebih dalam tentang teknik pengelolaan relasi, seperti `FetchType.LAZY` dan `FetchType.EAGER` seperti pada Gambar 4. 4, serta kapan masing-masing teknik ini harus diterapkan. Pada proyek ini, misalnya, penulis perlu memutuskan apakah harus menggunakan *fetch* yang *lazy* atau *eager* bergantung pada kasus penggunaan tertentu untuk menjaga efisiensi pemrosesan data.

```
@Entity
@Table(name = "meeting_room", schema = "public")
@Data
@Builder
@AllArgsConstructor
@RequiredArgsConstructor
public class MeetingRoom {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```

@Column(name = "nama")
private String nama;

@Column(name = "image_url")
private String imageUrl;

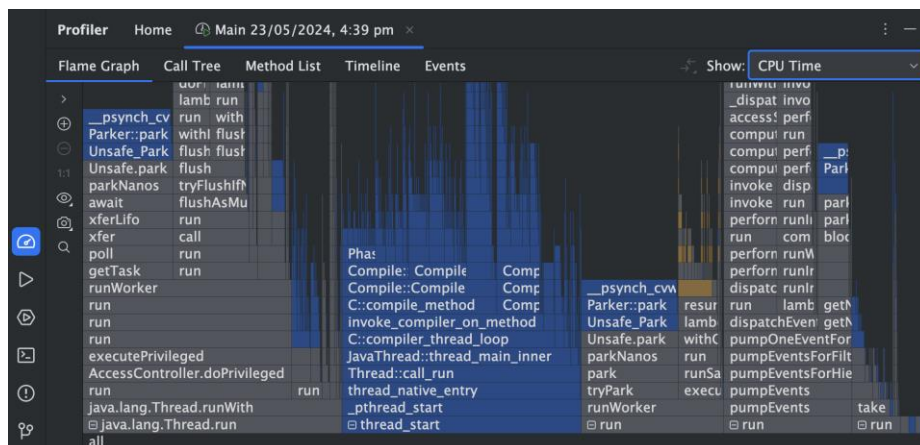
@Column(name = "kapasitas")
private int kapasitas;

@OneToMany(mappedBy = "ruangMeeting", cascade = CascadeType.ALL, fetch =
FetchType.LAZY)
private List<MeetingRoomFacility> meetingRoomFacilities;
}

```

Gambar 4. 4 Entitas MeetingRoom menggunakan metode *fetch* Lazy

Pengalaman di lapangan juga menunjukkan bahwa meskipun Spring Data JPA sangat membantu dalam mengurangi kode *boilerplate*, pengembang tetap harus berhati-hati dalam mengoptimalkan *query* yang dihasilkan oleh JPA. *Query* yang dihasilkan secara otomatis tidak selalu optimal, terutama untuk data yang memiliki volume besar. Dalam beberapa kasus, penulis harus melakukan *profiling* dan analisis performa *query* seperti pada Gambar 4. 5 untuk mengidentifikasi *query* mana yang membutuhkan optimasi agar tidak menyebabkan *overhead* yang signifikan pada sistem.



Gambar 4. 5 Profiling aplikasi Spring menggunakan IntelliJIdea

Secara keseluruhan, gap yang ditemukan dalam penggunaan Spring Data JPA ini memberikan pelajaran penting bahwa modul-modul otomatis dalam pengembangan aplikasi harus digunakan dengan pemahaman mendalam terkait keterbatasan dan konsekuensinya. Penulis juga belajar bahwa pengelolaan relasi, optimasi *query*, serta pengendalian *fetching* sangat krusial untuk memastikan bahwa aplikasi dapat berjalan dengan performa optimal, khususnya ketika bekerja dengan data dalam jumlah besar. Dengan demikian, kontribusi akademik dari pengalaman ini adalah pentingnya memahami arsitektur dan perilaku *query* yang dihasilkan oleh JPA agar sistem tetap efisien dan reliabel.

Spring Cloud GCP

Spring Cloud GCP digunakan dalam proyek pengembangan aplikasi BSI PINTER untuk mengintegrasikan penyimpanan file melalui Google Cloud Storage. Spring Cloud GCP memungkinkan pengembang untuk menggunakan layanan Google Cloud dengan cara yang lebih sederhana dan terintegrasi, dengan menyediakan konfigurasi otomatis untuk akses ke *bucket* penyimpanan dan operasi pengelolaan file. Teknologi ini dirancang untuk mengurangi kompleksitas dalam pengembangan dan meminimalkan kode *boilerplate*, yang berarti pengembang seharusnya dapat mengakses layanan *cloud* tanpa perlu melakukan konfigurasi manual yang rumit.

Namun, dalam praktiknya, penulis menemukan sejumlah perbedaan dan tantangan ketika menerapkan Spring Cloud GCP di lapangan. Salah satu tantangan utama adalah pengelolaan kredensial akun layanan (*service account credentials*). Teori menunjukkan bahwa penggunaan kredensial akun layanan hanya memerlukan konfigurasi sederhana dalam file aplikasi, tetapi dalam implementasi sebenarnya, penulis dihadapkan pada kompleksitas pengelolaan kredensial yaitu aspek keamanan. Kredensial harus dijaga keamanannya dan harus dapat diakses dengan benar oleh aplikasi tanpa menimbulkan risiko keamanan.

Selain itu, penggunaan Spring Cloud GCP untuk mengelola penyimpanan *file* memerlukan optimasi tertentu dalam hal pengelolaan performa. Teori menyatakan bahwa Spring Cloud dapat secara otomatis menangani koneksi ke Google Cloud Storage, tetapi dalam praktiknya, penulis harus menambahkan logika tambahan untuk menangani penamaan file dan optimasi penyimpanan. Misalnya, penulis menambahkan mekanisme penamaan unik untuk *file* menggunakan *RandomString* untuk menghindari konflik nama *file* dalam *bucket* yang sama. Penulis juga menemukan bahwa pengelolaan validasi tipe file menjadi penting untuk memastikan hanya tipe *file* yang diizinkan yang dapat diunggah, yang membutuhkan penanganan manual tambahan, mengingat tidak semua fungsi ini disediakan langsung oleh Spring Cloud.

Dari pengalaman ini, penulis menyimpulkan bahwa penerapan Spring Cloud GCP dalam praktiknya memerlukan lebih banyak penyesuaian dan pemahaman mendalam tentang bagaimana mengelola keamanan kredensial, optimasi performa, serta penanganan *file* yang sesuai dengan kebutuhan bisnis. Hal ini menunjukkan bahwa meskipun teori menyatakan Spring Cloud GCP dapat mempermudah integrasi *cloud*, kenyataannya, pengembang harus

memiliki keterampilan teknis untuk mengatasi masalah yang muncul selama integrasi, seperti pengelolaan autentikasi yang aman dan penghindaran konflik *file*.

4.1.2 Penerapan Metode Agile dengan Scrum

Penerapan metode Scrum yang digunakan dalam proyek BSI PINTER kurang lebih diimplementasikan sesuai panduan dari Scrum Guide (Schwaber & Sutherland, 2020). Pengembangan dengan metode Scrum dilakukan secara iteratif dan inkremental, dengan tujuan untuk menghasilkan sebuah fitur (*increment*) dalam waktu yang cepat, fleksibel dan adaptif terhadap perubahan. Dalam teori, Scrum menekankan pentingnya transparansi, inspeksi, dan adaptasi, serta pelaksanaan *event-event* seperti *Sprint Planning*, *Daily Scrum*, *Sprint Review*, dan *Sprint Retrospective*. Setiap peran dalam Scrum memiliki peran penting dalam memastikan kelancaran proyek dalam artian setiap fitur dikembangkan secara maksimal.

Penggunaan Jira

Selama pelaksanaan magang, penulis menggunakan Jira untuk mengembangkan aplikasi *backend* menggunakan metode Scrum. Dalam penerapannya, penulis merasakan bahwa fitur-fitur dalam Jira sangat memenuhi kebutuhan yang cukup kompleks dari aplikasi BSI PINTER. Dalam Jira, tim dapat berkomunikasi secara efektif dengan backlog yang diberikan secara detail. Pada fitur backlog di Jira, seluruh kebutuhan terdefinisi secara jelas dan detail. Pada konteks QA, penulis merasakan bahwa penggunaan Jira memungkinkan tim developer dan tim QA berjalan secara selaras, setiap bug dari hasil tes dapat terdefinisi dengan jelas dan dikelola secara baik.

Pada waktu kuliah, penulis sempat mengerjakan beberapa proyek akhir untuk mata kuliah. Penulis memiliki inisiasi untuk menggunakan Jira dalam mengelola proyek tersebut. Berdasarkan pengalaman penulis selama magang, Jira memiliki beberapa fitur yang memang ditujukan untuk proyek yang memiliki kompleksitas tinggi. Salah satunya adalah dalam pendefinisian backlog, dapat dirancang secara detail dan pada setiap backlognya dapat memiliki dependensi satu sama lain. Penulis merasa bahwa kompleksitas fitur Jira sangat memenuhi kebutuhan yang diinginkan oleh proyek BSI PINTER. Namun, pada praktiknya, dalam pengerjaan proyek mata kuliah penulis merasa bahwa aplikasi Jira memiliki cakupan fitur yang cukup kompleks dalam menangani proyek sekelas proyek akhir mata kuliah. Meskipun demikian, fitur dalam Jira masih dapat secara efektif digunakan untuk menangani proyek dengan skala kecil.

Menurut Penulis, penggunaan Jira masih cukup relevan dalam proyek mata kuliah meskipun proyek tersebut memiliki cakupan yang kecil. Jira masih dapat secara efektif digunakan dengan menggunakan fitur-fitur yang esensial sesuai dengan kebutuhan minimal yang diharapkan. Pengguna tidak harus menggunakan seluruh fitur yang disajikan oleh Jira untuk mendapatkan dampak yang diharapkan. Penggunaan Jira memungkinkan tahap perencanaan yang efisien, terdapat template Scrum yang dapat digunakan secara langsung sehingga mempercepat proses pengembangan.

Sprint Planning

Secara teori, Sprint Planning harus melibatkan seluruh anggota tim, termasuk *stakeholder*, untuk menentukan *backlog* dan *task* yang akan dikerjakan selama Sprint. Namun, dalam praktiknya, penyesuaian yang dilakukan pada pengembangan proyek seringkali lebih fleksibel dan melibatkan diskusi terbatas hanya pada *developer*, *Product Owner* dan *Product Manager*. *Stakeholder* tidak selalu terlibat secara langsung, sehingga *backlog* yang dipilih untuk dikerjakan terkadang lebih fokus pada aspek teknis daripada kebutuhan bisnis. Hal ini tercermin pada *backlog* yang dapat berubah ubah dalam setiap fase Sprint. Sesuai dengan prinsip Scrum bahwa tim Scrum harus adaptif terhadap perubahan, diskusi yang intensif dapat membantu anggota tim untuk tetap selaras dengan Sprint Goal.

Daily Scrum

Secara teori, *Daily Scrum* berfungsi untuk melacak perkembangan pekerjaan dan mengidentifikasi hambatan yang muncul selama proses pengembangan. Namun, pada praktiknya, durasi *Daily Scrum* terkadang memerlukan waktu yang lebih daripada waktu yang telah dilaksanakan. Tidak jarang dengan waktu yang cukup singkat beberapa masalah yang dihadapi tidak teridentifikasi dengan baik. *Daily Scrum* perlu dilakukan proses diskusi secara intensif dengan berfokus pada kemajuan setiap anggota tim khususnya kendala yang dihadapi selama proses pengembangan. Proses penentuan durasi *Daily Scrum* perlu dilakukan secara teliti dengan menganalisa kebutuhan durasi *Daily Scrum* terhadap durasi pada fase Sprint.

Sprint Review Meeting

Secara teori, dalam pelaksanaan Sprint Review Meeting mengharuskan tim untuk mendemonstrasikan hasil increment dari Sprint yang telah dilaksanakan. Tim juga diharuskan untuk melakukan validasi terhadap Sprint goals yang telah ditetapkan. Dalam pelaksanaannya,

stakeholders dapat memberikan feedback dari hasil demonstrasi increment. Event ini merupakan kesempatan bagi tim untuk melakukan inspeksi dan adaptasi terhadap backlog berdasarkan kebutuhan terkini.

Pada praktiknya, dalam pelaksanaan magang, penulis melaksanakan event ini dengan mendemonstrasikan fitur yang telah dikembangkan. Dalam konteks *backend*, increment yang didemonstrasikan adalah hasil *endpoint* yang telah dikembangkan selama fase Sprint. Tim melakukan validasi hasil increment yang telah dihasilkan sesuai dengan Sprint goals yang ditetapkan. Stakeholders mengumpulkan feedback yang didapatkan dari hasil demonstrasi. Dalam praktiknya, tim magang mendemonstrasikan hasil increment berdasarkan role masing-masing. Tim android, mendemonstrasikan hasil UI yang telah dikembangkan yang telah diintegrasikan dengan *backend*. Pada event ini, tim dan product owner juga mendiskusikan perubahan prioritas backlog jika diperlukan. Pada tahap akhir, product owner melakukan konfirmasi increment, apakah increment sudah sesuai dengan *acceptance criteria* yang telah didefinisikan. Jika belum sesuai, maka increment yang belum sesuai dengan kriteria akan dilanjutkan pada Sprint Selanjutnya.

Sprint Retrospective Meeting

Secara teori, dalam pelaksanaan Sprint Retrospective Meeting, tim diharuskan untuk melakukan refleksi terhadap seluruh proses yang telah berlangsung selama Sprint. Event ini bertujuan untuk mengevaluasi apa yang berjalan dengan baik, apa yang perlu diperbaiki, dan apa yang bisa ditingkatkan untuk Sprint berikutnya. Dalam pelaksanaannya, seluruh anggota tim berbagi pandangan mereka secara terbuka mengenai pengalaman selama Sprint, termasuk kendala yang dihadapi dan solusi yang bisa diterapkan. Event ini menjadi kesempatan formal bagi tim untuk meningkatkan efektivitas kerja dengan mendiskusikan area yang dapat ditingkatkan dan menyepakati rencana aksi yang konkret untuk penerapan di Sprint selanjutnya. Hal ini dilakukan sebagai wujud adaptasi tim untuk mencapai hasil yang lebih optimal di masa mendatang.

Dalam praktiknya, pada pelaksanaan event ini penulis melaksanakan event sesuai teori yang dijelaskan. Dalam pelaksanaan event ini tim BSI PINTER menggunakan software Miro untuk melakukan refleksi terhadap seluruh proses yang telah berlangsung selama sprint. Dalam waktu yang ditentukan (kurang lebih 10 menit) seluruh anggota tim memberikan feedback dalam bentuk poin-poin singkat dalam sticky notes yang ditambahkan pada setiap bagian evaluasi yaitu apa yang telah berjalan dengan baik, apa yang perlu diperbaiki, dan apa yang

bisa ditingkatkan pada Sprint berikutnya. Poin meliputi masukan teknis maupun non-teknis, seperti proses kolaborasi antar anggota tim, komunikasi, efisiensi penyelesaian tugas, dan pemanfaatan *tools* yang digunakan selama Sprint. Setelah waktu untuk memberikan feedback selesai, setiap anggota tim melakukan diskusi bersama melakukan elaborasi untuk membahas setiap poin yang telah ditambahkan. Diskusi ini bertujuan untuk memahami lebih dalam konteks dari setiap poin feedback, mengidentifikasi akar permasalahan, dan menentukan langkah konkret yang dapat diambil untuk memperbaiki proses kerja di Sprint selanjutnya. Hasil dari diskusi ini kemudian dirangkum menjadi rencana aksi yang spesifik yang akan menjadi acuan bagi tim dalam meningkatkan produktivitas dan efisiensi pada Sprint berikutnya.

Validasi dan Testing

Secara teori, fitur yang dikembangkan selama fase Sprint harus memenuhi *acceptance criteria* dan siap untuk diimplementasikan atau diuji. Pada praktiknya beberapa fitur belum melalui proses testing secara penuh selama Sprint, dan proses pengujian baru dilakukan setelah Sprint selesai. Proses pengujian yang dilakukan terkadang juga tidak selaras dengan skenario yang telah ditetapkan diawal. Hal ini menunjukkan bahwa dalam praktiknya beberapa fitur mungkin dapat menghadapi keterbatasan waktu atau sumber daya, sehingga proses testing tidak selalu dapat berjalan secara paralel dengan pengembangan fitur. Pengelolaan waktu dan prioritas *backlog* harus diperhatikan dalam pelaksanaan Scrum untuk memastikan fitur diuji secara tuntas selama Sprint.

4.2 Pembelajaran Magang

Magang adalah salah satu tahap penting dalam perjalanan akademik mahasiswa yang bertujuan untuk mengembangkan keterampilan yang dibutuhkan dalam dunia profesional. Pengalaman magang tidak hanya menjadi sarana untuk belajar secara langsung, tetapi juga berperan sebagai persiapan yang menyeluruh dalam menghadapi tuntutan industri di masa mendatang. Melalui kegiatan magang, mahasiswa dapat menerapkan pengetahuan teoritis yang didapatkan di perkuliahan ke dalam situasi nyata, sehingga memperkuat pemahaman sekaligus mengasah kemampuan praktis.

Selama program magang, penulis memperoleh manfaat dalam mengembangkan keterampilan teknis dan non-teknis. Keterampilan teknis terutama berfokus pada penerapan Spring Framework dalam pengembangan aplikasi *backend* berbasis REST API. Di sisi lain, keterampilan seperti manajemen waktu, komunikasi, dan kerja sama tim juga mengalami

peningkatan. Pengalaman ini memberikan wawasan tentang dinamika kerja industri dan membantu penulis beradaptasi dengan cepat terhadap perubahan. Dengan menggabungkan teori dari perkuliahan dan pengalaman langsung, magang ini mempersiapkan penulis untuk menghadapi tantangan di dunia kerja.

Keterampilan yang diperoleh selama magang mencakup pemahaman mendalam mengenai teknologi dan metodologi yang digunakan dalam proyek pengembangan aplikasi BSI PINTER. Pengalaman terlibat dalam pengembangan fitur-fitur utama menggunakan modul Spring seperti Spring Boot, Spring Security, dan Spring Data JPA, memberikan wawasan praktis tentang bagaimana teori perkuliahan diterapkan dalam konteks nyata. Selain itu, penulis juga mempelajari bagaimana metode *agile*, khususnya Scrum, diterapkan untuk memastikan pengembangan perangkat lunak berjalan efisien dan terstruktur.

Di luar keterampilan teknis, magang juga memberikan kesempatan untuk mengembangkan *soft skills* yang sangat penting dalam dunia kerja profesional, seperti kemampuan komunikasi, kerja sama tim, dan manajemen waktu. Melalui keterlibatan dalam tim yang beragam, penulis belajar untuk berkomunikasi secara efektif, baik dalam menyampaikan ide maupun menerima *feedback*. Keterampilan ini semakin diperkuat melalui kolaborasi tim, yang menjadi salah satu fondasi utama keberhasilan dalam menyelesaikan proyek-proyek pengembangan yang kompleks.

BAB V

PENUTUP

5.1 Kesimpulan

Pengembangan *backend* aplikasi BSI PINTER menggunakan Spring Framework telah selesai dilakukan dan berhasil mencapai tujuan utama, yaitu menyediakan solusi digital terintegrasi untuk memudahkan akses informasi dan meningkatkan efisiensi operasional di Bank Syariah Indonesia. Sistem yang dikembangkan mampu mengelola fitur kuis dan pengelolaan ruang *meeting* secara efektif dengan menggunakan REST API sebagai antarmuka integrasi ke aplikasi frontend. Pengujian yang dilakukan menunjukkan bahwa API yang dikembangkan dapat diakses dengan baik oleh aplikasi mobile, mendukung berbagai skenario penggunaan yang direncanakan. Pada konteks penggunaan metode Scrum, setiap *acceptance criteria* pada *backlog* yang didefinisikan oleh sudah terpenuhi dengan baik berdasarkan hasil review oleh *stakeholder*.

Penerapan Spring Framework terbukti mendukung skalabilitas dan keamanan aplikasi, sementara integrasi metode Scrum dalam pengembangan menghasilkan proses yang adaptif dan terstruktur. Hasil dari proyek ini berhasil menyediakan *backend* yang fungsional dan siap diintegrasikan oleh tim frontend developer untuk digunakan oleh pengguna akhir, sesuai dengan tujuan pengembangan aplikasi BSI PINTER. Pengalaman ini memberikan wawasan mendalam dalam penerapan teknologi dan metodologi pengembangan perangkat lunak secara nyata, serta relevansi yang signifikan dengan teori yang telah dipelajari selama studi.

5.2 Saran

Terdapat saran bagi pengembangan aplikasi BSI PINTER di fase Sprint selanjutnya. Pada fitur kuis, hasil pengembangan aplikasi sudah memenuhi kebutuhan yang telah ditentukan. Beberapa skenario juga sudah diuji dan berjalan dengan baik. Namun, hasil evaluasi kuis masih kurang dalam memberikan analisa secara mendalam, diperlukan fitur tambahan seperti analitik atau statistik yang dapat memberikan wawasan yang lebih detail mengenai performa pada masing-masing individu. Fitur tersebut dapat membantu manajemen dalam memantau kemajuan pembelajaran pada karyawan secara lebih detail, seperti misalnya tren skor, waktu penyelesaian, dan area yang memerlukan peningkatan. Dengan hal itu, manajemen dapat dengan mudah untuk mengambil keputusan yang lebih tepat berdasarkan data kuantitatif yang dihasilkan.

Pada fitur pengelolaan ruang *meeting*, meskipun fungsi dasar seperti pemesanan ruang *meeting* dan pengaturan jadwal sudah berfungsi sesuai dengan kebutuhan, tetapi masih terdapat ruang untuk peningkatan. Pada Sprint selanjutnya, product owner dapat menambahkan kebutuhan untuk validasi yang ketat terkait pemesanan ruang *meeting*. Mekanisme lain yang dapat ditambahkan adalah dalam hal perubahan pesanan atau pembatalan penggunaan ruang *meeting*. Proses tersebut juga harus melalui validasi yang ketat untuk memastikan tidak terjadi konflik jadwal penggunaan. Pihak pengembang juga harus memastikan transaksi berjalan dengan konsisten, memastikan setiap perubahan atau pembatalan diproses secara atomik dan konsisten, sehingga meminimalisasi risiko inkonsistensi data. Selain itu, pengujian secara intensif juga diperlukan pada skenario penggunaan yang nyata untuk memastikan sistem berfungsi dengan baik.

Pada fase Sprint selanjutnya, penerapan metode Scrum dapat ditingkatkan, terutama dalam hal perencanaan serta eksekusi. Meskipun Scrum sudah berjalan sesuai dengan teori, beberapa penyesuaian perlu dilakukan untuk memastikan bahwa *backlog* yang dipilih lebih sesuai dengan prioritas bisnis dan teknis. Sebagai contoh, pada fase Sprint planning, disarankan sebisa mungkin seluruh *stakeholder* lebih terlibat dalam penentuan *prioritas backlog*, sehingga fitur yang dikembangkan benar-benar mencerminkan kebutuhan strategis perusahaan. Selain itu, penggunaan metode estimasi yang efektif juga diperlukan, seperti penentuan *story points* berbasis kompleksitas perlu dioptimalkan untuk memastikan Sprint diselesaikan sesuai target.

DAFTAR PUSTAKA

- Bauer, C., King, G., & Gregory, G. (2016). *JAVA PERSISTENCE WITH HIBERNATE* (Second Edition). Shelter Island, New York: Manning Publications.
- Bonteanu, A. M., & Tudose, C. (2024). Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA, Hibernate, Spring Data JPA. *Applied Sciences (Switzerland)*, 14(7). <https://doi.org/10.3390/app14072743>
- Cosmina, I., Harrop, R., Schaefer, C., & Ho, C. (2017). *Pro Spring 5* (Fifth Edition). Berkeley, CA: Apress. <https://doi.org/10.1007/978-1-4842-2808-1>
- Craig, W. (2016). *Spring Boot In Action* (First Edition). Shelter Island, New York: Manning Publications.
- Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Dissertations). University of California, Irvine.
- Fisher, P., & Murphy, B. D. (2016). Spring Persistence with Hibernate, Second Edition. In *Spring Persistence with Hibernate, Second Edition*. Apress Media LLC. <https://doi.org/10.1007/978-1-4842-0268-5>
- Gajewski, M., & Zabierowski, W. (2019). Analysis and Comparison of the Spring Framework and Play Framework Performance, Used to Create Web Applications in Java. *2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, 170–173. IEEE. <https://doi.org/10.1109/MEMSTECH.2019.8817390>
- Geewax, J. (2018). *Google Cloud Platform IN ACTION* (First Edition). Shelter Island: Manning Publications.
- Geoff Watts. (2013). *Scrum Mastery: From Good to Great Servant-Leadership* (First edition). Cheltenham, UK: Inspect & Adapt Ltd.
- Hubli, S. C., & Jaiswal, Dr. R. C. (2023). Efficient Backend Development with Spring Boot: A Comprehensive Overview. *International Journal for Research in Applied Science and Engineering Technology*, 11(11), 759–765. <https://doi.org/10.22214/ijraset.2023.56617>
- Islam, M., Rahaman, S., Meng, N., Hassanshahi, B., Krishnan, P., & Yao, D. D. (2020). Coding Practices and Recommendations of Spring Security for Enterprise Applications. *Proceedings - 2020 IEEE Secure Development, SecDev 2020*, 49–57. Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/SecDev45635.2020.00024>

- Jin, B., Sahni, S., & Shevat, A. (2018). *Designing Web APIs BUILDING APIS THAT DEVELOPERS LOVE* (First Edition). Sebastopol: O'Reilly Media.
- Ken, A. (2023, October 4). Backend: What is it and what is it used for? Retrieved May 19, 2024, from <https://www.gluo.mx/en-US/blog/backend-que-es-y-para-que-sirve>
- Knutson, M., Winch, R., & Mularien, P. (2017). *Spring Security* (Third Edition). Birmingham: Packt Publishing.
- Madurapperuma, I., Shafana, M., & Sabani, M. (2022). *State-of-Art Frameworks for Front-end and Back-end Web Development*.
- Red Hat. (2020, May 8). What is a REST API? Retrieved May 19, 2024, from <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- Richardson, L., Amundsen, M., & Ruby, S. (2013). *RESTful Web APIs* (First). Sebastopol: O'Reilly Media.
- Roynardo, R., & Bello, F. E. (2017, October 4). Framework Spring Java. Retrieved May 19, 2024, from <https://socs.binus.ac.id/2017/10/04/framework-spring-java/>
- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*.
- Spilca, L. (2020). *Spring Security IN ACTION* (First Edition). Shelter Island, New York: Manning Publications.
- Spring. (n.d.). Introduction to Spring Framework. Retrieved August 3, 2024, from <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html>
- Sutherland, J. (2014). *More Praise for Scrum: The Art of Doing Twice the Work in Half the Time* (First Edition). New York: Crown Business.
- Walls, C. (2019). *Spring In Action* (fifth edition). Shelter Island, New York: Manning Publications.

LAMPIRAN