PENGEMBANGAN REST API SISTEM UIIADMISI DENGAN MICROSERVICES

Disusun Oleh:

N a m a : Muhammad Fachri Ramadhan
NIM : 19523197

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN REST API SISTEM UIIADMISI DENGAN MICROSERVICES

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Muhammad Fachri Ramadhan

NIM : 19523197

Yogyakarta, 12 Oktober 2023

Pembimbing,

(Zainudin Zukhri, S.T., M.IT.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN REST API SISTEM UIIADMISI DENGAN MICROSERVICES

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, 12 Oktober 2023

Tim Penguji

Zainudin Zukhri, S.T., M.IT. _____

Anggota 1

Kholid Haryono, S.T., M.Kom. _____

Anggota 2

Andhika Giri Persada, S.Kom., M.Eng. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

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

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Muhammad Fachri Ramadhan

NIM : 19523197

Tugas akhir dengan judul:

PENGEMBANGAN REST API SISTEM UIIADMISI DENGAN MICROSERVICES

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, 12 Oktober 2023

(Muhammad Fachri Ramadhan)

HALAMAN PERSEMBAHAN

Halaman ini penulis dedikasikan untuk kepada dua orang hebat dalam hidup saya, ayahanda dan ibunda yang selalu memberi semangat dan mendukung saya sehingga saya bisa sampai menyelesaikan skripsi ini. Persembahan juga ditujukan kepada adik saya yang selalu memberikan dukungan. Selain itu, skripsi ini penulis dedikasikan juga kepada sahabat-sahabat yang selalu memberikan bantuan dan dukungan.

HALAMAN MOTO

“Berakit rakit ke hulu berenang-renang ketepian”

"Barang siapa keluar untuk mencari sebuah ilmu maka ia akan berada di jalan Allah hingga ia kembali”

- HR Tirmidzi –

"Sesungguhnya Allah tidak akan mengubah nasib suatu kaum sehingga mereka mengubah keadaan yang ada pada diri mereka sendiri."

- Q.S. Ar-Ra’d Ayat 11 -

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah, puji syukur kita panjatkan ke hadirat Allah subhanahu wa ta'ala yang telah memberikan banyak nikmat sehingga penulis dapat menyusun skripsi ini dengan baik. Skripsi ini berisi tentang kegiatan magang yang dilakukan di Badan Sistem Informasi Universitas Islam Indonesia sebagai pengembang perangkat lunak. Skripsi ini dibuat untuk memenuhi persyaratan lulus dalam menjalani kuliah sarjana selama empat tahun. Penulis mengucapkan terima kasih yang sebesar-besarnya kepada pihak yang telah mendukung penulis untuk menyelesaikan laporan ini.. Selain itu, bagian kata pengantar juga dapat memuat berbagai hal sebagai berikut:

1. Kedua orang tua dan keluarga yang selalu memberikan dukungan dan doa untuk selalu sukses.
2. Bapak Zainudin Zukhri, S.T., M.IT. selaku dosen pembimbing yang bersedia untuk membimbing penulis untuk dapat menulis laporan ini.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku ketua program studi sarjana Informatika Universitas Islam Indonesia.
4. Bapak Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D. selaku kepala Badan Sistem Informasi Universitas Islam Indonesia.
5. Kak Luthfi Anggy Kurniawan S.Kom. beserta seluruh anggota tim Admisi yang telah membina penulis secara teknis untuk dapat mengerjakan tugas yang diberikan.

Selama penyusunan skripsi ini, penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna. Kritik dan saran akan sangat berharga. Semoga skripsi ini dapat memberikan manfaat yang baik untuk penulis dan para pembaca.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 12 Oktober 2023

(Muhammad Fachri Ramadhan)

SARI

Program Penerimaan Mahasiswa Baru merupakan sebuah aktivitas yang dilakukan oleh sebuah institusi pendidikan perguruan tinggi untuk menerima mahasiswa baru yang dilakukan setiap tahun ajaran baru. Sistem UIIAdmisi adalah sebuah sistem informasi administrasi berbasis web yang digunakan untuk mendigitalisasikan pelaksanaan program penerimaan mahasiswa baru di Universitas Islam Indonesia. Sistem ini telah digunakan oleh penyelenggara program Penerimaan Mahasiswa Baru Universitas Islam Indonesia dari tahun 2016. Selama masa penggunaannya, pengembang baru merasakan kesulitan dalam mengembangkan sisi *back-end* sistem UIIAdmisi. Salah satu alasan mengapa sistem UIIAdmisi sulit untuk dikembangkan lebih lanjut oleh pengembang baru adalah karena UIIAdmisi menyelesaikan proses bisnis yang kompleks, namun sistem dibangun di atas arsitektur *monolith* sehingga membuat UIIAdmisi sulit untuk di-*scale* lebih jauh. Untuk itu, dilakukan proses migrasi arsitektur sistem UIIAdmisi dari *monolith* menuju *microservices* dengan memisahkan komponen yang ada pada sistem menjadi komponen kecil yang independen. Komponen-komponen kecil ini akan berkomunikasi melalui protokol HTTP dengan menerapkan antarmuka REST API. Sistem juga akan dibuat dengan menggunakan pendekatan *domain driven design* dan *clean architecture* agar kode pada fitur yang berada di sisi pendaftar menjadi lebih mudah untuk dikembangkan oleh pengembang baru. Hasil dari pengembangan ini adalah beberapa fitur di sisi pendaftar pada UIIAdmisi yang sudah digunakan oleh *end-user* saat ini, yaitu fitur beli formulir, *list* pendaftaran, *list* registrasi, isi data formulir, rekomendasi dan tagihan. Fitur-fitur tersebut dikembangkan menggunakan *clean architecture* dengan membuat komponen-komponen yang ada di dalamnya menjadi lebih modular sehingga fitur-fitur tersebut menjadi lebih mudah untuk dikembangkan lebih lanjut.

Kata kunci: website admisi, *domain driven design, microservices*, *back-end*, *scrum*.

GLOSARIUM

API	Antarmuka program yang menjadi pintu untuk mengakses program.
Agile	Metode pengembangan perangkat lunak dengan tempo cepat.
Applet	Aplikasi kecil yang dirancang untuk dijalankan dalam aplikasi lain.
Back-end	Sisi pada pengembangan web yang tidak dilihat oleh pengguna.
Constraint	Batasan.
Class	Cetak biru untuk objek pada pemrograman.
Database	Komponen untuk menyimpan data.
Deploy	Proses merilis sistem atau aplikasi.
Endpoint	Alamat pintu masuk sebuah API.
Error	Pengguna yang akan menggunakan sistem atau aplikasi.
Feedback	Saran dan kritik.
Framework	Kerangka kerja.
Front-end	Sisi pada pengembangan web yang hasilnya akan dilihat oleh pengguna.
Landing Page	Halaman yang pertama kali muncul ketika mengakses suatu *website*.
Maintain	Proses untuk memelihara sebuah perangkat lunak.
Merge	Proses untuk menggabungkan perubahan pada kode ke kode induk.
Microservice	Sebuah *web service* yang berada di lingkungan *microservices*.
Response	Respon dari server untuk klien.
Request	Klien meminta informasi ke server.
Techstack	Kombinasi teknologi yang digunakan.
Tools	Alat-alat pembantu.
Webservice	Layanan yang dapat diakses melalui *internet*.

DAFTAR ISI

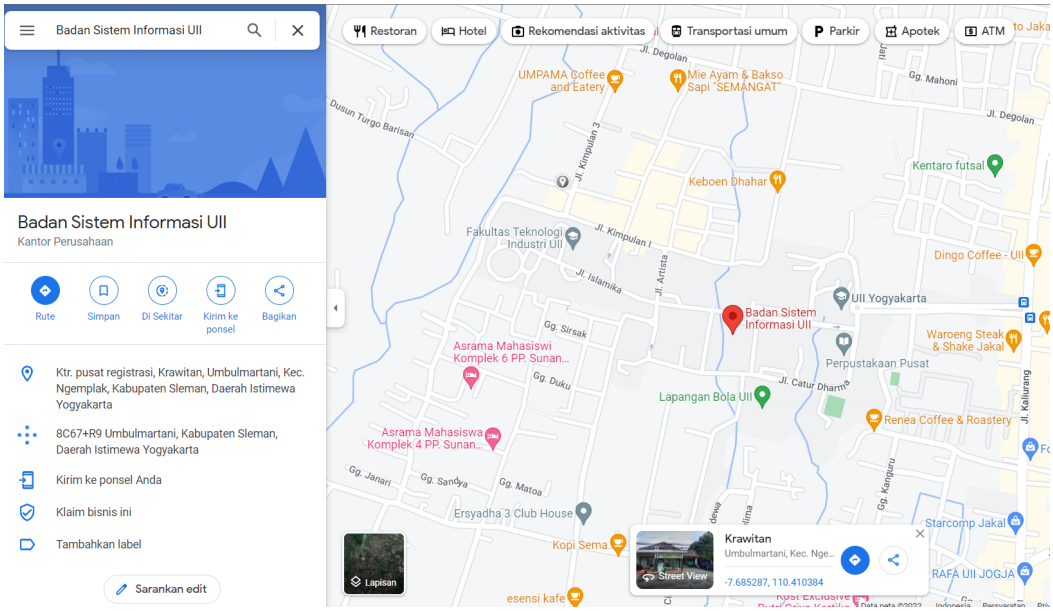
DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Badan Sistem Informasi Universitas Islam Indonesia (BSI UII) merupakan badan yang bergerak pada bidang sistem dan teknologi informasi di dalam lingkungan Universitas Islam Indonesia (UII). Peran Badan Sistem Informasi Universitas Islam Indonesia sangatlah besar sebagai badan yang melayani, mendampingi, dan mengakselerasi berbagai macam aspek di dalam lingkungan Universitas Islam Indonesia. Badan Sistem Informasi Universitas Islam Indonesia berperan dalam mengawal perencanaan, pengembangan, operasi, serta layanan sistem dan teknologi informasi di lingkungan UII. Kantor Badan Sistem Informasi Universitas Islam Indonesia berada di Gedung GPBH Prabuningrat lantai 4, seperti pada Gambar 1.1. Badan Sistem Informasi Universitas Islam Indonesia telah mengembangkan banyak sistem informasi dan aplikasi yang membantu banyak pihak di lingkungan UII, salah satunya adalah sistem UIIAdmisi yang digunakan oleh panitia Penerimaan Mahasiswa Baru.



Gambar 1.1 Lokasi kantor Badan Sistem Informasi Universitas Islam Indonesia

UIIAdmisi adalah sebuah sistem informasi berbasis web yang digunakan untuk membantu pelaksanaan program Penerimaan Mahasiswa Baru (PMB) di UII yang dilakukan setiap tahun. Sistem ini akan mengawali pendaftar dalam melakukan proses pendaftaran PMB UII, mulai dari proses membeli formulir pendaftaran hingga mengawali proses registrasi data

pendaftar ke sistem akademik. Selain mengawali pendaftar, sistem ini juga akan membantu di sisi panitia/petugas PMB pada saat melakukan seleksi kandidat pendaftar dan verifikasi data pendaftar.

UIIAdmisi telah dikembangkan dan digunakan oleh panitia PMB dari tahun 2016. Sistem UIIAdmisi dibangun dengan menerapkan arsitektur *monolith* dan masih belum menerapkan teknologi *framework*. Namun seiring dengan berkembangnya infrastruktur teknologi yang ada di Badan Sistem Informasi Universitas Islam Indonesia, UIIAdmisi perlu dikembangkan ulang mengikuti infrastruktur terbaru yang Badan Sistem Informasi Universitas Islam Indonesia telah terapkan. Seluruh sistem informasi dan aplikasi baru yang dikembangkan oleh Badan Sistem Informasi Universitas Islam Indonesia sudah mulai menerapkan arsitektur *microservices* dan memiliki standar yang harus dicapai sebelum aplikasi/sistem siap digunakan oleh *end-user* agar dapat memberikan aplikasi/sistem yang berkualitas dan mudah untuk dikembangkan lebih lanjut. Pada proyek pengembangan ulang sistem UIIAdmisi ini menargetkan sistem dapat mulai digunakan sebelum program PMB di UII mulai berjalan, yaitu pada tanggal 20 Desember 2022.

Laporan akhir ini akan menjelaskan bagaimana proses pengembangan ulang sistem UIIAdmisi pada sisi *back-end*. *Back-end* sistem UIIAdmisi akan dibuat ulang di atas Bahasa pemrograman PHP dan menggunakan *framework* Laravel Lumen untuk dapat membuat sebuah *web service*. *Back-end* UIIAdmisi juga akan menerapkan arsitektur *microservices*, membuat *back-end* UIIAdmisi menjadi komponen-komponen kecil yang dapat berkomunikasi dengan menggunakan protokol *Hypertext Transfer Protocol* (HTTP) melalui antarmuka *Representational State Transfer* (REST).

1.2 Ruang Lingkup

Pelaksanaan magang di Badan Sistem Informasi Universitas Islam Indonesia berlangsung selama delapan bulan dimulai dari awal September 2022 hingga akhir April 2023. Badan Sistem Informasi memiliki sekitar 80 karyawan yang dibagi menjadi tiga bidang, yaitu bidang perencanaan, bidang operasional dan bidang pengembangan. Di bidang pengembangan terdapat lima tim yang bekerja untuk mengembangkan sistem atau aplikasi pada ranah bisnis spesifik, salah satunya adalah tim Admisi. Tim Admisi menerapkan *scrum* sebagai *Software Development Life Cycle (*SDLC), Tim Admisi memiliki sebanyak 11 anggota yang di dalamnya terdapat beberapa peran, yaitu *product owner* , *scrum master*, *UI/UX designer*, *front-end,* dan *back-end developer*. Aktivitas-aktivitas yang dilakukan selama magang sebagai berikut:

a. Mengikuti program induksi karyawan SPK baru dan magang kerja.
b. Mengembangkan *back-end* sistem UIIAdmisi yang meliputi beberapa *web services*.

1.3 Tujuan

Adapun tujuan dari pelaksanaan magang di Badan Sistem Informasi Universitas Islam Indonesia, yaitu mengembangkan ulang sisi *back-end* sistem informasi UIIAdmisi dengan menggunakan arsitektur *microservices* dan pendekatan *domain driven design* serta konsep *clean architecture* agar UIIAdmisi menjadi lebih mudah untuk dipelihara.

1.4 Manfaat

Manfaat yang didapat dari pengembangan ulang sisi *back-end* sistem UIIAdmisi ini adalah sebagai berikut:

a. Membuat UIIAdmisi menjadi lebih mudah untuk dirawat dan dikembangkan lebih lanjut oleh *developer* baru.
b. Menyediakan dokumentasi API yang lengkap.

1.5 Sistematika Penulisan

Dalam penulisan laporan akhir ini, sistematika penulisan dibagi menjadi lima bab sebagai berikut:

a. BAB I PENDAHULUAN berisikan latar belakang, ruang lingkup, tujuan dan manfaat magang serta sistematika penulisan pada laporan ini.
b. BAB II LANDASAN TEORI DAN TINJAUAN PUSTAKA menguraikan teori-teori, definisi-definisi dan informasi dari berbagai referensi atau buku yang berkaitan dengan topik skripsi.
c. BAB III PELAKSANAAN MAGANG menjelaskan secara rinci terkait metodologi magang yang dilakukan beserta dokumentasi pelaksanaan magang.
d. BAB IV REFLEKSI PELAKSANAAN MAGANG menjelaskan analisis terkait teori-teori/kajian akademik yang telah diuraikan pada BAB II dengan pelaksanaan di lapangan selama magang dan hal-hal baru yang didapatkan selama magang.
e. BAB V PENUTUP berisikan kesimpulan yang menjawab apakah tujuan dari laporan ini tercapai atau tidak dan saran.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

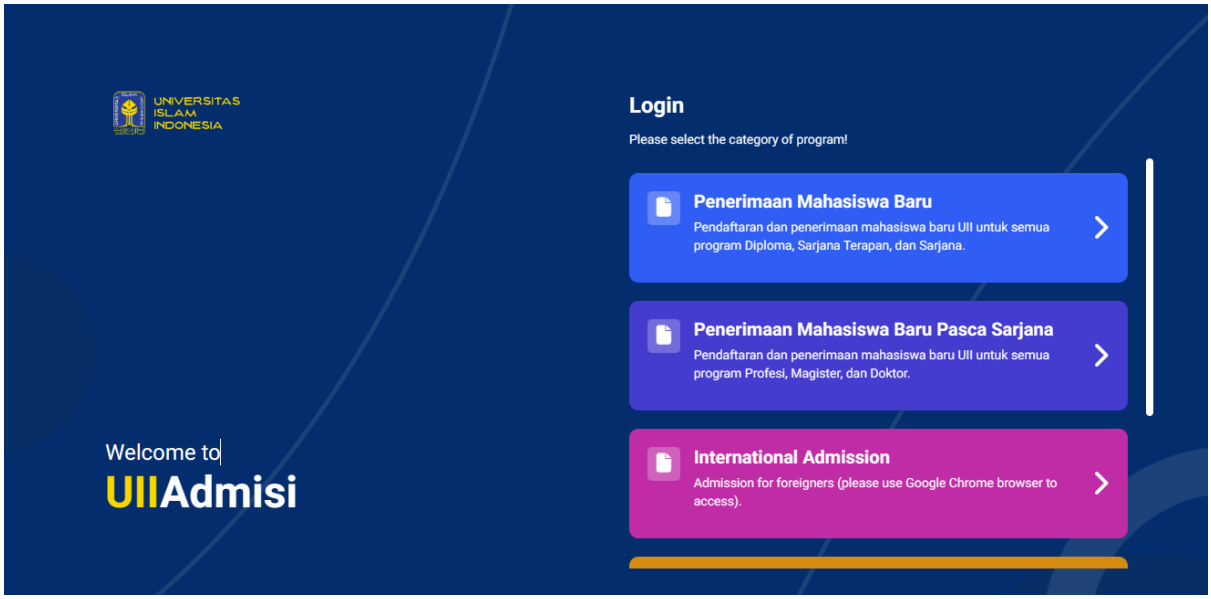
2.1 Sistem Informasi Admisi

Sistem informasi menurut (Leitch, R. A., & Davis, K. R., 1992) adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial serta strategis dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. Menurut KBBI, admisi adalah izin masuk, dalam konteks lembaga perguruan tinggi admisi adalah proses penerimaan mahasiswa, staf, dan tenaga kependidikan baru untuk masuk ke dalam lembaga perguruan tinggi tersebut. Dari kedua definisi tersebut dapat diartikan bahwa Sistem Informasi Admisi adalah sebuah sistem informasi yang mempertemukan kebutuhan pada proses penerimaan mahasiswa, staf, dan tenaga kependidikan baru dengan mendukung operasi, strategi, transaksi harian, dan menyediakan laporan-laporan yang diperlukan untuk pihak yang bersangkutan dengan admisi di lembaga tersebut.

2.2 UIIAdmisi

UIIAdmisi adalah sebuah sistem informasi admisi berbasis web yang digunakan untuk membantu pelaksanaan program PMB Universitas Islam Indonesia yang dilakukan setiap tahun. Penggunaan sistem informasi ini dapat menyederhanakan banyak aspek dalam proses admisi, seperti pengumpulan data, verifikasi, dan pengolahan dokumen. Pendaftar dapat mengisi formulir aplikasi secara *online* dari mana saja dan kapan saja. Mereka juga dapat mengakses informasi tentang jadwal, persyaratan, dan tahapan proses admisi secara *online*. Sistem ini dikembangkan dan dikelola oleh Badan Sistem Informasi Universitas Islam Indonesia. Pada program PMB Universitas Islam Indonesia terdapat beberapa grup pola seleksi yang dilaksanakan, yaitu Seleksi Berbasis Rapor (SIBER), *Computer Based Test* (CBT), *Paper Based Test* (PBT), dan Penelusuran Siswa Berprestasi (PSB). Masing-masing grup pola seleksi tersebut akan terdapat beberapa jenis pola seleksi yang memiliki proses bisnis tersendiri sehingga membuat sistem menjadi kompleks, seperti grup pola seleksi CBT terdapat jenis pola seleksi yang melakukan tes di UII dan ada yang melakukan tes di lembaga mitra. UIIAdmisi dapat diakses di https://admisi.uii.ac.id/landing. Gambar 2.1 merupakan halaman *landing page* UIIAdmisi. Pada UIIAdmisi, pengguna dapat melakukan pendaftaran untuk menjadi

mahasiswa sarjana, pasca sarjana, mahasiswa internasional, tenaga kependidikan, dan kredit transfer.



Gambar 2.1 *Landing page* UIIAdmisi

Pada sistem ini terdapat beberapa jenis pengguna yang nantinya akan menggunakan sistem ini, masing-masing pengguna tersebut akan memiliki peran masing-masing di dalam ekosistem UIIAdmisi. Jenis-jenis dari pengguna pada sistem UIIAdmisi, yaitu pendaftar, petugas dan kepala program studi. Sisi pendaftar dapat diakses melalui website https://admisi.uii.ac.id/landing, sedangkan sisi petugas dan kepala program studi diakses melalui UIIGateway. Pengguna pendaftar akan dapat mengikuti proses pendaftaran di pola seleksi PMB yang sudah dibuka, proses pendaftaran tersebut dimulai dari membuat akun Nomor Induk Utama (NIU) hingga melakukan registrasi ulang. Petugas merupakan pihak dari sisi UII yang akan mengawasi dan mengatur pelaksanaan seleksi PMB, seperti melakukan verifikasi terhadap data berkas yang dikirim oleh pendaftar. Kepala program studi memiliki peran sebagai pihak yang mengatur skor minimal yang harus dicapai untuk dapat diterima pada masing-masing program studi di UII serta menentukan jumlah kuota calon mahasiswa yang dapat diterima pada program studi tersebut. Grup pola seleksi beserta pola seleksi di dalamnya yang akan diimplementasikan ke dalam UIIAdmisi adalah sebagai berikut:

a. Seleksi Berbasis Rapot (SIBER), pada grup pola seleksi ini terdapat beberapa jenis pola seleksi di dalamnya, yaitu:
    1. SIBER juara.

2. SIBER reguler.
3. SIBER kedokteran.

b. *Computer Based Test* (CBT), pada grup pola seleksi ini terdapat beberapa jenis pola seleksi di dalamnya, yaitu:
   1. CBT kampus UII.
   2. CBT mitra.
   3. CBT kedokteran.

c. Penelurusan Siswa Berprestasi (PSB), pada grup pola seleksi ini terdapat beberapa jenis pola seleksi di dalamnya, yaitu:
   1. PSB hafizah hafiz.
   2. PSB duafa.
   3. PSB santri.
   4. PSB atlet & juara seni.
   5. PSB KIP.

d. *Paper Based Test* (PBT), pada grup pola seleksi ini hanya terdapat satu jenis pola seleksi di dalamnya, yaitu PBT Mitra.

2.3 Microservices

Microservices merupakan sebuah arsitektur perangkat lunak yang memecahkan kompleksitas pada sistem menjadi lebih kecil dengan cara memisahkan aplikasi menjadi sejumlah *web service* kecil yang independen. Masing-masing *web service* akan menangani tugas tertentu dan dapat berkomunikasi dengan menggunakan protokol *Hyper Text Transfer Protocol* (HTTP) melalui antarmuka *Application Programming Interface* (API), seperti REST dan SOAP atau melalui *Remote Procedure Call* (RPC). Arsitektur *microservices* memungkinkan untuk dikembangkan dengan teknologi yang berbeda pada setiap *web service* nya.

Arsitektur *microservices* telah banyak diterapkan pada aplikasi dan sistem yang berskala besar, seperti Gojek, Grab, Tokopedia dan Shopee yang menangani *data traffic* yang besar serta harus melayani user yang banyak di setiap harinya. Dengan memecahkan kompleksitas sistem menjadi komponen yang lebih kecil, sistem akan menjadi lebih mudah untuk dikembangkan lebih lanjut dan dapat dibuat menjadi lebih optimal jika dibandingkan saat menggunakan arsitektur *monolith*. Penggunaan arsitektur *microservices* membawa beberapa keunggulan yang tidak didapat pada arsitektur *monolith*, seperti proses pengembangan

perangkat lunak yang lebih cepat, dapat menggunakan *techstack* yang berbeda-beda (Sinambela, Ernawati, Coastera, 2021), kemampuan sistem untuk ditingkatkan menjadi lebih baik, mendukung *error isolation*, sistem lebih mudah untuk dirawat, dan membuat sistem menjadi lebih komprehensif (Nadareushvuku, Mitra, Mclarty, Amundsen, 2016).

2.4 Domain Driven Design

Domain-Driven Design adalah sebuah pendekatan dalam pengembangan perangkat lunak yang berfokus pada pemahaman mendalam tentang domain bisnis yang akan dikerjakan. Domain bisnis akan dianggap sebagai inti dari aplikasi dan desain perangkat lunak ditujukan untuk mencerminkan struktur dan konsep yang ada dalam domain tersebut.

Konsep bisnis tersebut akan diabstraksi oleh *domain expert* dan tim teknis menjadi sebuah model. Model ini akan menjadi sebuah jembatan komunikasi atau *ubiquitous language* yang membantu dalam pemahaman yang akurat tentang konsep-konsep, aturan, dan proses dalam domain bisnis, serta memfasilitasi komunikasi yang lebih baik di antara semua pemangku kepentingan (Suryotrisongko, 2017). Penggunaan *ubiquitous language* ini tidak hanya diterapkan hanya pada saat berdiskusi dengan *domain expert* saja, namun juga diterapkan pada saat menulis kode, seperti penamaan *class* dan *function*. Dengan mengimplementasikan pendekatan *domain driven design* pengembangan sistem yang kompleks dapat menjadi lebih mudah untuk dipahami (Nuralamsyah, Akbar, Fabroyir, 2021). Di dalam implementasi *tactical design domain driven design* terdapat beberapa *building blocks*, yaitu:

a. *Domain model* merupakan representasi formal dari domain bisnis dalam perangkat lunak yang mencakup objek, atribut, dan metode. *Domain model* juga akan memetakan bagaimana hubungan antar objek tersebut.
b. *Entities* adalah objek yang memiliki identitas unik dalam domain bisnis. Atribut pada *entities* dapat berubah.
c. *Value Object* adalah objek yang memiliki nilai atau atribut unik.
d. *Aggregate* adalah kelompok objek terkait yang membentuk suatu kesatuan tunggal yang konsisten dalam domain bisnis. Objek pada *Aggregate* dapat berupa *value object* atau *entities*.
e. *Domain service* merupakan komponen yang di dalamnya terdapat logika bisnis pada suatu *domain model*.

f. *Repositories* merupakan komponen yang menjadi *data access layer* dengan komponen *database*. Komponen ini memungkinkan sistem untuk melakukan operasi *Create*, *Read*, *Update*, *Delete* (CRUD) pada suatu *entities*.

2.5 Clean Architecture

Clean Architecture merupakan sebuah prinsip arsitektur sistem yang memecah fungsional di dalamnya menjadi beberapa *layer* yang memiliki fungsional tersendiri untuk meningkatkan *maintainability* sebuah sistem. Pemisahan fungsional menjadi beberapa *layer* memiliki tujuan untuk *separation of concerns* sehingga masing-masing *layer* memiliki satu tujuan tertentu. *Clean Architecture* juga akan membuat sistem menjadi *loosely coupled* yang artinya komponen yang ada pada sistem dapat dipisah dan dipasang tanpa menyebabkan *error* yang banyak selama komponen tersebut memiliki fungsional yang sama. *Layer* yang ada pada *clean architecture* sebagai berikut:

a. *Application Layer*, pada *layer* ini terdapat *use case* yang di dalamnya terdapat sebuah fungsi logika bisnis dari satu atau banyak *entities*.
b. *Domain Model Layer*, pada *layer* ini terdapat *entities* dan *value object* dari domain bisnis yang terdiri dari atribut dan *behavior*.
c. *Infrastructure Layer*, *layer* ini digunakan untuk mengakses layanan ketiga yang digunakan oleh sistem, seperti basis data dan layanan *cloud*.
d. *Interface Layer*, *layer* ini berguna untuk mengatur bagaimana klien dapat mengakses sistem.

2.6 REST

Representational State Transfer atau REST adalah gaya arsitektur web yang biasanya digunakan untuk mempermudah komunikasi di antara *web service* dengan menyediakan standar komunikasi melalui HTTP. Gaya arsitektur REST ini dikemukakan oleh Roy Thomas Fielding. REST merupakan gabungan dari beberapa gaya arsitektur berbasis web (Fielding, 2000) yang menerapkan beberapa *constraint* pada gaya arsitektur tersebut, *constraint* tersebut sebagai berikut:

a. *Client-Server, constraint* ini menjelaskan bahwa *client* dan *server* diimplementasikan secara independen sehingga implementasi pada sisi *client* dapat berubah tanpa mengganggu sisi *server* dan begitu juga sebaliknya.

b. *Stateless,* komunikasi yang terjadi di antara klien dan server harus bersifat *stateless* yang artinya setiap *request* yang dikirim harus memiliki informasi yang dibutuhkan tanpa menggunakan *context* yang disimpan pada sisi server.
c. *Cache, constraint cache* mendefinisikan untuk setiap data pada sebuah *response* dari sebuah *request* untuk dilabelkan sebagai *cacheable* atau *non-cacheable*. Pengimplementasian *constraint* ini dapat meningkatkan efisiensi namun memiliki kekurangan ketika data yang telah di *cache* dapat menjadi berbeda dengan data yang dikirimkan oleh server ketika *request* secara langsung.
d. *Uniform Interface, constraint* ini mendefinisikan dengan penerapan prinsip *generality* pada rekayasa perangkat lunak, arsitektur sistem dapat disimplifikasi dan meningkatkan visibilitas interaksi sistem. Hal ini tercapai karena empat *constraint* yang diterapkan pada *interface,* yaitu identifikasi *resources*, *resources* dapat dimanipulasi melalui representasi, pesan yang dapat mendeskripsikan diri, dan *hypermedia* sebagai mesin status aplikasi.
e. *Layered System, constraint* ini memungkinkan sebuah sistem menjadi tersusun dari lapisan hierarkis yang membatasi komponen di setiap lapisan. Lapisan dapat digunakan untuk mengenkapsulasi layanan lama dan menyederhanakan komponen dengan memindahkan fungsionalitas yang jarang digunakan ke perantara bersama.
f. *Code-On-Demand, constraint* ini mendefinisikan REST memungkinkan untuk menambahkan fungsionalitas pada *client* dengan mengunduh dan mengeksekusi kode dalam bentuk *applet* atau *script.*

RESTful API adalah antarmuka perangkat lunak yang dibuat menerapkan prinsip REST. RESTful API dapat melakukan komunikasi ke sistem lain dengan menggunakan protokol HTTP melalui HTTP *request* lalu sistem yang menerima HTTP *request* tersebut akan menjawab dengan memberikan HTTP *response* yang berisikan sebuah *resource. Resource* yang diberikan oleh sebuah *RESTful API* dapat berupa dalam bentuk *Extensible Markup Language* (XML) atau *Javascript Object Notation* (JSON).

Pada HTTP *request* terdapat beberapa komponen yang harus ada untuk melakukan HTTP *request*, yaitu *request line*, *request header*, dan *message body*. *Request line* berisikan HTTP *method,* seperti GET, POST, PUT, DELETE, UPDATE, alamat atau *Unique Resource Identifier* (URI), dan versi protokol HTTP yang digunakan. *Request header* berisikan informasi tambahan tentang *request* dan *client* yang melakukan *request*, seperti autentikasi, *host*, tanggal *request*, dan lain-lain. *Message body* merupakan komponen opsional yang dapat memiliki isi

berupa data terkait yang dibutuhkan pada *request* tertentu, seperti data pada sebuah formulir pendaftaran.

HTTP *response* juga memiliki beberapa komponen pada di dalamnya, yaitu *status line*, *response headers*, dan *message body*. *Status line* berisikan beberapa informasi yang mendeskripsikan hasil dari *request*, terdapat pesan dan *status code* yang mengindikasikan status keberhasilan dari *request*, *status code* yang umum muncul adalah 200 untuk *request* yang berhasil, 400 untuk kesalahan yang terdapat pada sisi *client*, dan 500 untuk kesalahan yang terjadi pada server. *Response header* mirip dengan *request header,* yaitu berisikan informasi tambahan mengenai *response*. *Message body* berisikan sebuah *resource* yang diminta oleh klien (Mozilla Developer).

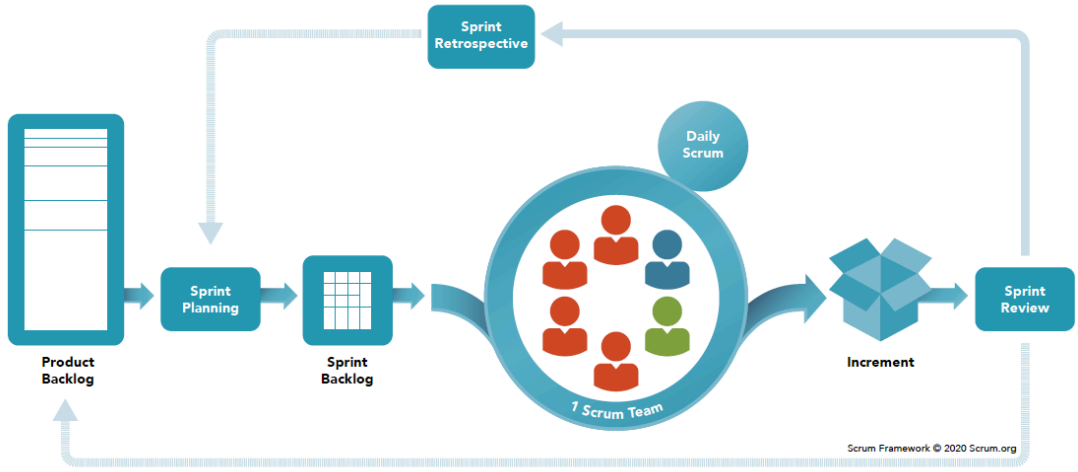
2.7 Scrum

Scrum adalah suatu kerangka kerja yang digunakan dalam pengembangan perangkat lunak yang mengadopsi pendekatan *agile*. Dalam *scrum*, pengembangan perangkat lunak dibagi menjadi serangkaian iterasi pendek yang disebut *sprint*. Setiap *sprint* memiliki tujuan yang jelas dan waktu yang terbatas, biasanya *sprint* berjalan sekitar 1 hingga 4 minggu. Dalam *scrum*, tim pengembang bekerja secara kolaboratif dan terstruktur. Tim terdiri dari anggota tim yang meliputi *scrum master*, *product owner*, dan anggota pengembang, seperti *UI/UX designer*, *back-end developer*, *front-end developer*, *Ops*, dll (Hadji, Taufik, Mulyono, 2019). *Scrum master* bertanggung jawab untuk memastikan penerapan *scrum* yang efektif, memfasilitasi komunikasi tim, dan menghilangkan hambatan yang menghalangi kemajuan proyek. *Product owner* adalah pemangku kepentingan yang bertanggung jawab untuk mengatur prioritas fitur dan kebutuhan pelanggan serta memastikan nilai bisnis dari produk yang dikembangkan.

Scrum menggunakan *artifact,* seperti *product backlog*, *sprint backlog*, dan *product increment* untuk membantu pengelolaan proyek. *Product backlog* adalah daftar fitur dan kebutuhan yang perlu dikembangkan, sedangkan *sprint backlog* adalah daftar tugas yang harus diselesaikan dalam *sprint* tertentu. *Product increment* adalah *product backlog* yang sudah diselesaikan pada *sprint* yang berlangsung (Harris).

Salah satu aspek penting dalam *scrum* adalah *scrum events*. *Scrum events* ini dilakukan agar transparansi dapat dilihat oleh semua anggota tim. *Scrum events* terdiri dari *daily scrum*, *sprint*, *sprint planning*, *sprint review*, dan *sprint retrospective* (Bauroziq, 2022). Gambar 2.2 merupakan gambaran dari berjalannya sebuah *sprint*. Pada *daily scrum*, anggota tim berbagi informasi tentang pekerjaan yang telah selesai, pekerjaan yang sedang dilakukan, dan

hambatan yang dihadapi. Hal ini memungkinkan tim untuk tetap sinkron dan mengidentifikasi masalah yang perlu diselesaikan. *sprint* adalah periode tim bekerja untuk menyelesaikan *product backlog*. *Sprint planning* adalah pertemuan di awal *sprint* untuk menentukan tujuan dari *sprint* dan *sprint backlog* (Barozigq, 2022). *Sprint review* adalah pertemuan yang dilakukan di akhir *sprint,* tim akan melakukan peninjauan atas pengerjaan *sprint backlog* pada *sprint* itu. *Sprint retrospective* adalah pertemuan yang dilakukan setelah *sprint* selesai. Tim akan melihat kembali *sprint* yang telah selesai dan mengidentifikasi apa yang berhasil dan apa yang perlu diperbaiki untuk meningkatkan kinerja tim di masa depan.



Gambar 2.2 Alur *sprint* pada *scrum*

2.8 Laravel Lumen

Laravel Lumen adalah sebuah *framework* mikro berbasis bahasa pemrograman PHP yang dikembangkan oleh tim Laravel. Laravel Lumen dirancang dengan fokus pada kecepatan dan *scalability* yang tinggi, serta memberikan kemudahan dalam membangun aplikasi web yang ringan dan cepat. Laravel Lumen jika dibandingkan dengan Laravel merupakan versi yang lebih ringan dan lebih *optimized*. Dikarenakan kecepatan dan *scalability* yang tinggi, Laravel Lumen sering digunakan untuk membuat sebuah *web service* pada sebuah sistem berbasis *microservices*. Namun, Laravel Lumen mengorbankan beberapa fitur yang ada pada Laravel. Fitur utama yang terdapat pada Laravel Lumen adalah sebagai berikut:

a. *Routing*.
b. *Authentication.*
c. *Caching.*
d. *Logging.*

e. *Error Handling.*

f. *Query Builder.*

g. *Database Migration*, *Factories* dan *Seeder*.

h. *Model View Controller (MVC) Pattern*.

Selain fitur utama yang telah disebutkan, Laravel Lumen juga memiliki *package* tambahan yang dibuat oleh pembuat Laravel dan komunitas yang dapat menambahkan fungsional pada sebuah proyek Laravel Lumen, seperti Carbon, Eloquent *Object Relational Mapper* (ORM), dan Facades.

2.9 MySQL

MySQL adalah sebuah Relational Database Management System (RDBMS) yang sangat terkenal dan sering digunakan dalam pengembangan perangkat lunak. Mengikuti model Relational Database, MySQL memungkinkan pengguna untuk menyimpan, mengelola, dan mengakses data dalam bentuk tabel terstruktur yang memiliki hubungan satu sama lain melalui *foreign key*. Dalam konteks ini, MySQL menyediakan cara yang terstruktur untuk mengelola data. MySQL memiliki fitur *trigger* yang merupakan sebuah prosedur yang akan berjalan ketika terjadi manipulasi data pada tabel, fitur *trigger* ini bersifat *synchronous* yang mana prosedur akan langsung berjalan. Selain *trigger*, MySQL juga memiliki fitur *view* yang berfungsi untuk membuat sebuah tabel virtual untuk menampilkan data.

2.10 Tinjauan Pustaka

Terdapat beberapa penelitian yang telah dilakukan mengenai pengembangan RESTful API menggunakan arsitektur *microservices*, *domain driven design* dan *clean architecture*. Penelitian yang dilakukan oleh Fajri & Rani (2022) menerapkan *clean architecture* dan pola desain MVVM pada aplikasi Android. Kedua teori tersebut diterapkan dengan tujuan agar kode pada aplikasi menjadi terstruktur dan mudah dirawat. Penerapan *clean architecture* pada penelitian ini memisahkan fungsional kode menjadi tiga lapisan, yaitu *presentation layer*, *domain layer*, dan *data layer*. Hasil dari penggunaan *clean architecture* dan MVVM pada aplikasi membuat proses perawatan kode menjadi lebih mudah dan proses pengembangan menjadi lebih terstruktur.

Penelitian yang dilakukan oleh Pawana dkk. (2021) mendokumentasikan perubahan arsitektur perangkat lunak Sistem Informasi Dosen di Universitas Udayana dari arsitektur *monolithic* ke arsitektur *microservices* sebagai solusi untuk mempermudah pengembangan

sistem yang memiliki *codebase* yang besar dan meningkatkan *scalability* aplikasi. Hasilnya adalah sebuah sistem berbasis *microservices* yang dibuat dengan mengimplementasikan pendekatan *domain driven design* dan menggunakan RESTful API. *Services* yang terdapat pada sistem bersifat *independent* sehingga tidak mengganggu *services* lain secara signifikan.

Penelitian yang dilakukan oleh Hadji dkk. (2019) mendokumentasikan pengembangan aplikasi *delivery order* berbasis web dengan menggunakan *scrum* sebagai metode pengembangan perangkat lunak. Dari penelitian ini didapatkan bahwa metode *scrum* dapat menghadapi perubahan dalam *requirement* selama fase pengembangan sistem. Jika produk dalam *sprint* pertama belum sepenuhnya memenuhi kebutuhan, maka pada *sprint* berikutnya, sistem dapat dikembangkan sesuai berdasarkan umpan balik dari pengguna. Kesimpulan dari beberapa penelitian sebelumnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian relevan

No	Referensi	Metode Pengembangan	Tools / teori pendukung	Hasil
1.	Pawana, Wiharta & Sastra, 2021	Studi permasalahan, studi literatur, pengumpulan data, Analisa kondisi eksisting sistem, perancangan model *microservices*, evaluasi model	*Domain driven design*, *microservices*, REST API	Sistem berbasis *microservices* yang dibuat dengan mengimplementasikan pendekatan *domain driven design* dan menggunakan RESTful API.
2.	Hadji, Taufik & Mulyono, 2019	Tahap pengumpulan data, Analisa sistem berjalan, Tahap pengembangan sistem	*Agile*, *scrum*	Aplikasi *delivery order* berbasis web.
3.	Fajri & Rani, 2022	*Product backlog*, *sprint planning*, *sprint*, *daily standup*, *sprint review*, *sprint retrospective*	*Clean architecture*, *scrum*, MVVM, *android*	Aplikasi *mobile* Agree Partner berbasis Android

Berdasarkan pada Tabel 2.1, ketiga penelitian tersebut memiliki kesamaan, yaitu mengembangkan sebuah sistem berbasis web dengan metodologi, *tools* dan pendekatan yang berbeda-beda. Pada penelitian ini akan menggabungkan beberapa metode, *tools* dan teori yang ada pada Tabel 2.1 dalam mengembangkan RESTful API di dalam arsitektur *microservice* untuk sistem UIIAdmisi. Penelitian ini akan membahas bagaimana pengimplementasian pendekatan *domain driven design* dan *clean architecture* pada *microservice* berbasis REST yang dibuat menggunakan fitur yang ada di *framework* Laravel lumen dan menggunakan *scrum*.

BAB III
PELAKSANAAN MAGANG

3.1 Manajemen Proyek

Kegiatan magang di BSI UII dilaksanakan dalam salah satu tim pengembangan, yaitu tim Admisi yang terdiri dari 11 anggota yang di dalamnya terdapat beberapa peran, yaitu product owner , scrum master, UI/UX designer, front-end, dan back-end developer. Tim Admisi dapat dilihat pada Gambar 3.1. Kegiatan magang dilaksanakan dari hari Senin hingga hari Jumat. Pada awalnya jadwal jam kerja magang dimulai dari pukul 08.00 WIB hingga pukul 12.00 WIB. Setelah masuk ke Tim Admisi, jadwal jam kerja magang berubah mengikuti jadwal kerja Tim Admisi, yaitu mulai dari pukul 08.00 WIB hingga pukul 16.00 WIB dengan jam istirahat mulai dari pukul 12.00 WIB hingga pukul 13.00 WIB.

Gambar 3.1 Tim Admisi BSI UII

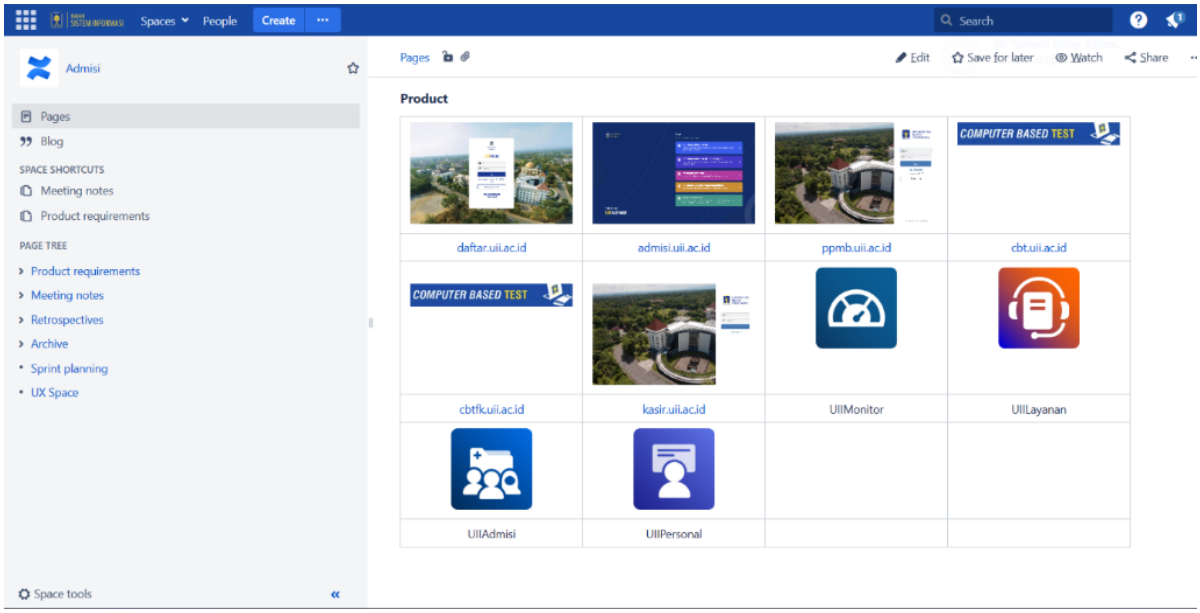
3.1.1 Tools

Dalam pengembangan UIIAdmisi, diterapkan metode siklus pengembangan perangkat lunak *agile* dengan *framework scrum* untuk mengatasi perubahan *requirement* atau prioritas proyek dengan baik. Dalam pelaksanaannya, tim Admisi menggunakan beberapa *tools* untuk

membantu manajemen proyek, monitor progres *sprint*, komunikasi antar anggota tim dan repositori kode sistem. Salah satu *tools* yang digunakan adalah Jira yang digunakan sebagai *task management tools* yang memudahkan tim dalam mengatur proyek dengan komponen, tugas, dan jadwal yang terstruktur dengan cara membuat daftar tugas, menetapkan prioritas, dan mengelola alur kerja proyek dengan efisien. Terdapat beberapa *task management tools* yang dapat digunakan selain Jira, seperti Taiga, ActiveCollab dan lain-lain. Alasan mengapa Jira digunakan pada proses pengembangan ini adalah Jira lebih berpusat pada pelacakan isu dan *bug* sehingga tim dapat melaporkan masalah tersebut selama proyek sedang berjalan (Ramadhini, Paputungan, 2022). Dengan Jira, tim dapat membuat dan melacak isu-isu yang meliputi *bug*, permintaan fitur, tugas lainnya dan *sprint backlog*. Setiap isu akan diberi deskripsi, lampiran, dan label, serta dapat dilengkapi dengan komentar dan pembaruan status yang membantu tim memantau perkembangan dari waktu ke waktu. Tim Admisi juga memberikan sebuah nilai kepada setiap isu dan *sprint backlog*, nilai yang diberikan merupakan bilangan *fibonacci* dari 1 hingga 23, nilai ini merepresentasikan beban kerja pada isu tersebut, nilai 1 berarti isu tersebut memiliki beban kerja yang ringan dan 23 berarti isu tersebut memiliki beban kerja yang berat. Jira juga menyediakan berbagai laporan dan metrik yang membantu tim Admisi untuk memantau dan menganalisis kinerja proyek. Laporan ini mencakup riwayat aktivitas, *burndown chart*, dan waktu yang dihabiskan pada pengerjaan *Sprint Backlog*. Data ini membantu tim dalam mengevaluasi efisiensi, mengidentifikasi hambatan, dan membuat perbaikan yang diperlukan.

Confluence digunakan untuk memfasilitasi kolaborasi tim sebagai *platform* kolaborasi dan berbagi pengetahuan yang memungkinkan anggota tim untuk membuat dan mengubah berbagai jenis dokumentasi, seperti dokumentasi hasil pengerjaan *backlog*, catatan hasil *meeting*, dokumentasi API, dan dokumentasi kebutuhan sistem untuk memungkinkan revisi dengan mudah. Salah satu alasan mengapa Confluence digunakan sebagai *tools* kolaborasi tim di BSI UII adalah Confluence dapat diintegrasikan dengan Jira sehingga dapat memungkinkan alur kerja yang efisien. Confluence juga menyimpan riwayat perubahan setiap dokumentasi untuk melihat perubahan yang telah dilakukan dan dapat mengembalikan ke versi sebelumnya jika diperlukan. Selain itu Confluence juga digunakan untuk berbagi dokumentasi dengan tim lain di BSI UII. Gambar 3.2 merupakan tampilan halaman Confluence yang digunakan oleh tim Admisi. Pada setiap akhir *sprint*, *scrum master* akan mengevaluasi kinerja antar anggota tim dan akan mendokumentasikan hasil evaluasi tersebut di Confluence. Kombinasi dari penggunaan Jira dan Confluence yang tepat mempermudah proses pembagian pengetahuan dan

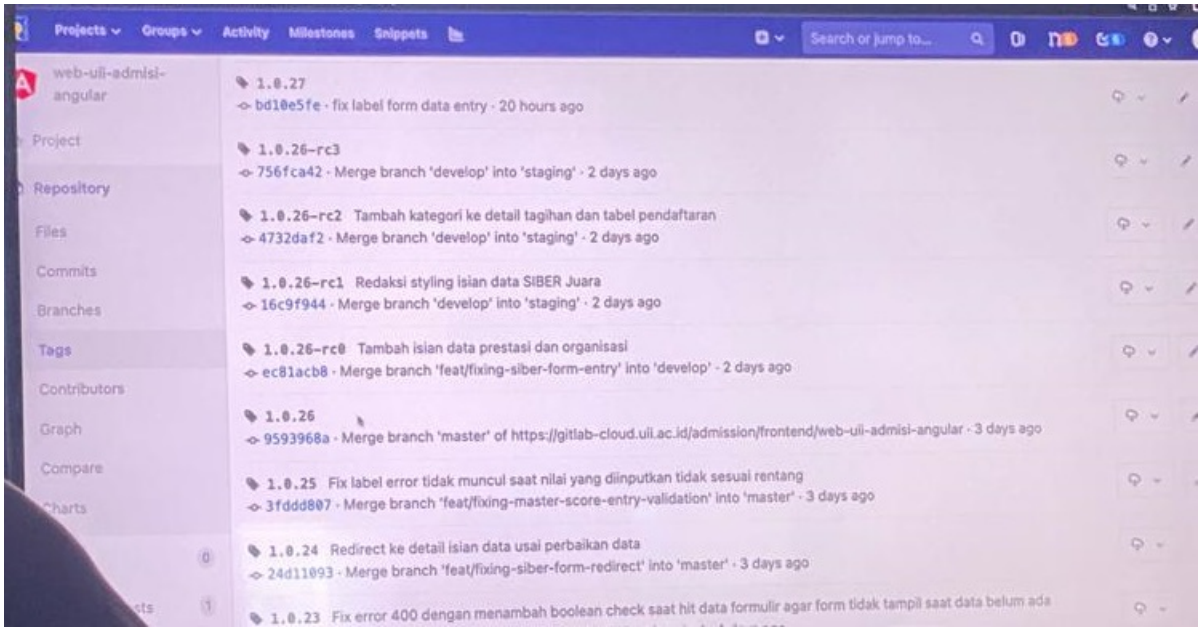
kolaborasi antar anggota tim maupun dari eksternal sehingga transparansi pada saat pengembangan sistem sangatlah tinggi.



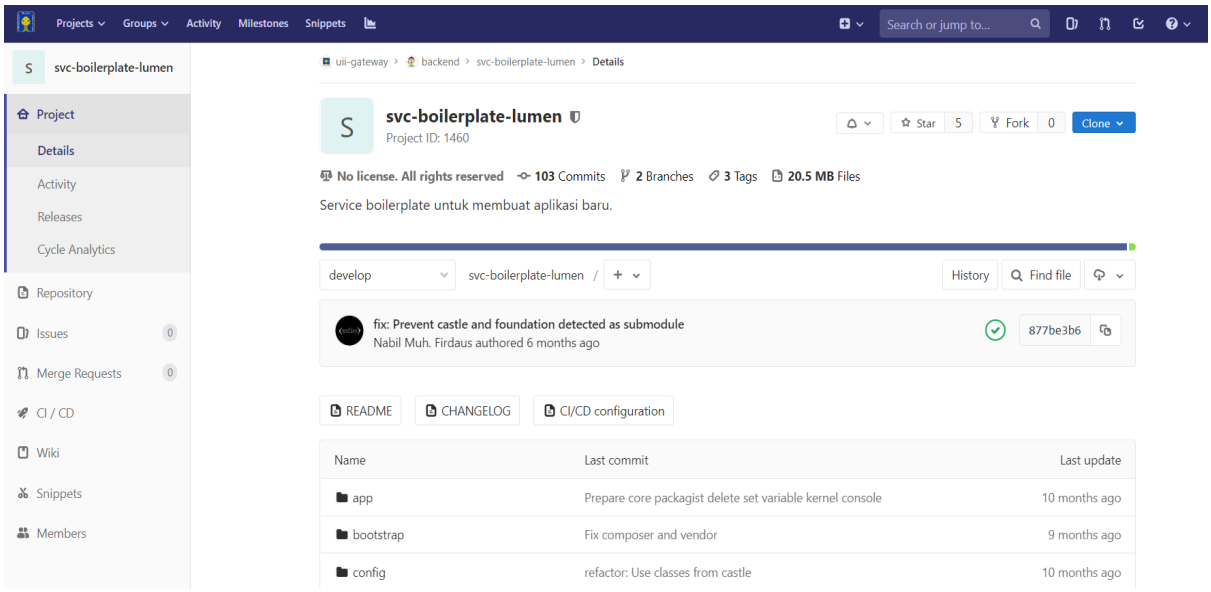
Gambar 3.2 Tampilan dari Confluence

Tim Admisi juga menggunakan Gitlab sebagai platform *repositories source code*. Gitlab juga memungkinkan *developer* untuk melakukan *version control* dengan melacak perubahan pada *source code* pada sebuah *microservice* yang sedang dikembangkan. Selain itu, Gitlab juga digunakan oleh tim admisi untuk membantu mengelola siklus pengembangan *microservice* dengan membuat tiga *branch* pada masing-masing *repositories*, yaitu *develop*, *staging*, dan *production*. *Branch develop* untuk mengembangkan fitur baru pada sebuah *microservice*, setelah mengembangkan fitur tersebut berikutnya *source code* tersebut akan di-*merge* ke *branch staging* untuk dilakukan tes oleh *UI/UX designer* yang juga bertugas untuk melakukan tes pada fitur baru di tim Admisi, jika *UI/UX designer* merasa fitur yang di tes sudah siap untuk digunakan oleh *end-user* baik dari segi fungsionalitas dan non-fungsionalitas, berikutnya fitur tersebut akan di-*merge* ke *branch production* dan aplikasi tersebut akan memiliki versi baru dengan menggunakan fitur *tag* pada Gitlab. Tim Admisi menerapkan *semantic versioning* dalam menamakan versi sistem untuk memberikan informasi tentang perubahan sistem secara jelas dan konsisten seperti pada gambar 3.3. *Semantic versioning* terdiri dari tiga urutan, yaitu *major*, *minor*, dan *patch*. *Major* akan bertambah ketika terdapat perubahan signifikan pada sistem yang merusak kompatibilitas dengan versi sebelumnya. *Minor* akan bertambah ketika terjadi perubahan yang tidak merusak kompatibilitas dan *patch* bertambah ketika terjadi perbaikan bug. Masing-masing *branch* tersebut memiliki *environment* yang terpisah dan

pipeline tersendiri yang mana di dalam *pipeline* tersebut dapat terjadi proses analisis *source code*, *build source code*, dan *deploy microservice*. Gambar 3.4 merupakan tampilan halaman *repositories* pada Gitlab.



Gambar 3.3 *Version control* pada Gitlab

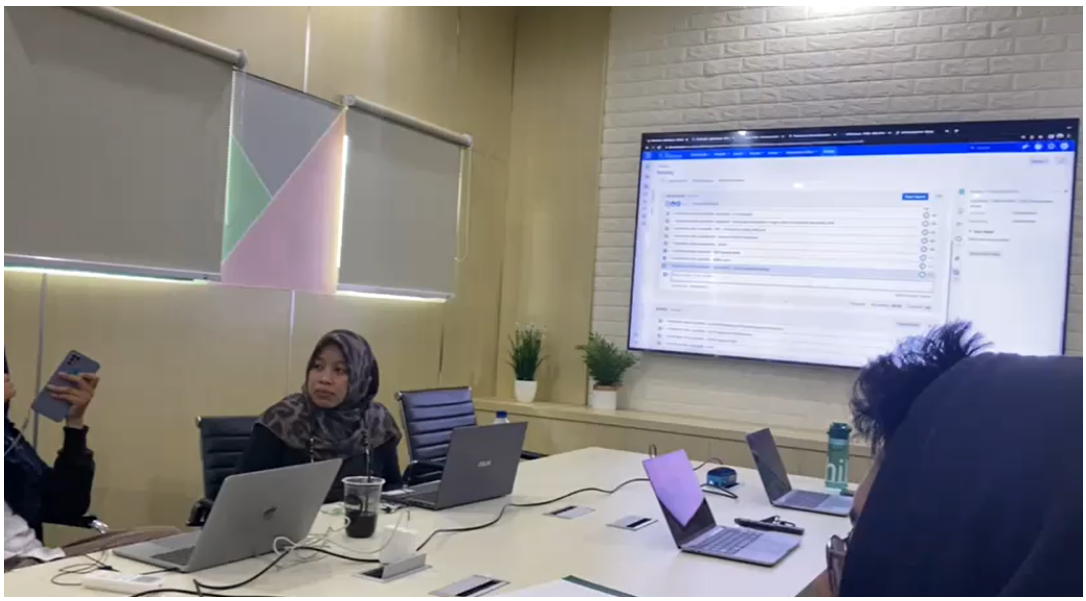


Gambar 3.4 Halaman salah satu *repositories* di Gitlab

3.1.2 Scrum

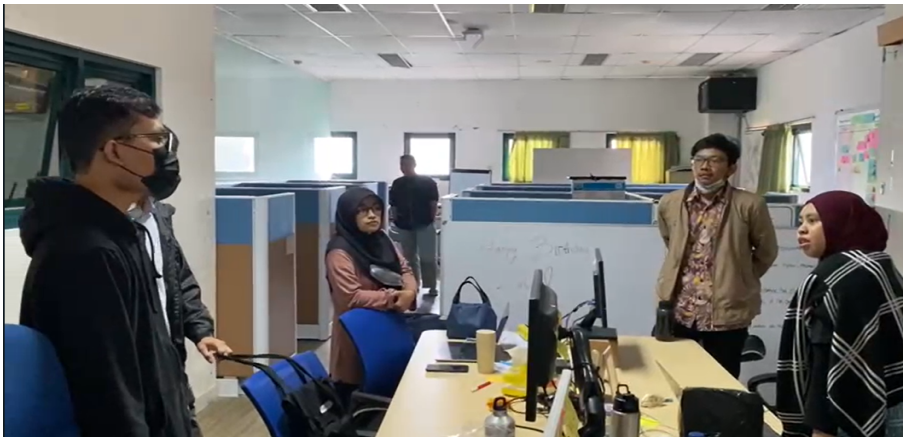
Dalam mengembangkan UIIAdmisi, tim Admisi menggunakan *scrum* sebagai metode siklus pengembangan perangkat lunak. Alasan mengapa metode *scrum* digunakan pada pengembangan adalah karena proses pengembangan pada scrum bersifat fleksibel dan adaptif yang artinya perubahan dapat diadopsi di tengah proses pengembangan berlangsung, berbeda dengan metode pengembangan lain, seperti *Waterfall* yang bersifat sistematis. Tim Admisi menerapkan seluruh *scrum events* mulai dari *sprint*, *sprint planning*, *daily scrum*, *sprint review*, dan *sprint retrospective*. Tim Admisi menetapkan untuk melaksanakan *sprint* selama dua minggu.

Setiap *sprint* akan dimulai dengan *sprint planning* untuk bertemu dan berdiskusi tentang *product backlog* mana saja yang akan dijadikan *sprint backlog* pada *sprint* tersebut dan bagaimana prioritas *sprint backlog* pada *sprint* itu. *Sprint planning* ini dihadiri oleh seluruh anggota tim mulai dari *product owner*, tim pengembang dan akan diawasi oleh seorang *scrum master* agar kegiatan *sprint planning* dapat berjalan dengan efektif. *Sprint* akan baru dimulai ketika tim telah menentukan *sprint backlog* beserta anggota mana yang akan mengerjakan *sprint backlog* tersebut, menentukan beban kerja pada *sprint backlog* tersebut, dan menentukan waktu *sprint* berakhir pada Jira. Gambar 3.5 merupakan gambaran dari kegiatan *sprint planning* yang dilakukan oleh tim Admisi.



Gambar 3.5 *Sprint planning* tim Admisi

Pada setiap hari di dalam periode *sprint* pada jam 09.00 WIB, tim Admisi akan melakukan *daily scrum*. Pada *daily scrum* masing-masing anggota pengembang akan menyampaikan *sprint backlog* apa saja yang dikerjakan di hari kemarin dan yang akan dikerjakan pada hari ini serta kendala yang dihadapi selama mengerjakan *sprint backlog* tersebut. Selain itu, *product owner* juga dapat memberikan informasi dan masukan terkait perubahan pada sistem yang diminta oleh *stakeholder* dan *scrum master* juga dapat memberikan informasi tambahan berupa informasi dari pihak manajemen. Gambar 3.6 adalah dokumentasi kegiatan *daily scrum* pada tim Admisi.



Gambar 3.6 Dokumentasi *daily scrum* tim Admisi

Pada akhir *sprint*, lebih tepatnya saat dua minggu setelah *sprint planning* tim Admisi akan melakukan *sprint review* untuk meninjau progres pengerjaan *sprint backlog* dan performa masing-masing anggota tim pengembang. Hasil dari *sprint review* ini dapat menjadi pertimbangan bagi *product owner* tim Admisi untuk menentukan prioritas pada *sprint backlog* di *sprint* berikutnya. Gambar 3.7 merupakan foto dokumentasi *sprint review* tim Admisi.

Gambar 3.7 Dokumentasi *sprint review* tim Admisi

Setelah melakukan *sprint review*, *sprint events* yang dilakukan adalah *sprint retrospective* dan menjadi kegiatan untuk mengakhiri *sprint* yang sedang berjalan. Pada *Sprint Retrospective* tim akan melakukan kegiatan rekreasi dan berdiskusi terkait performa pelaksaan *sprint* yang telah dilakukan serta memberi masukan kepada tim aspek apa saja yang dapat ditingkatkan agar *sprint* berikutnya dapat berjalan dengan efektif. Gambar 3.8 merupakan dokumentasi salah satu kegiatan *sprint retrospective* di tim Admisi.

Gambar 3.8 Dokumentasi *sprint retrospective* tim Admisi

3.2 Pelaksanaan Magang

Aktivitas magang telah dilaksanakan selama delapan bulan terhitung sejak 4 September 2022 hingga 30 April 2023 dengan memegang posisi sebagai *software engineer* di BSI UII. Di awal pelaksanaan magang, kegiatan magang adalah mengikuti program induksi SPK baru dan magang kerja untuk dapat mengenal lebih dalam tentang BSI UII beserta nilai-nilai yang dijunjung. Setelah mengikuti program induksi, kegiatan magang dilakukan dengan tim Admisi untuk mengerjakan proyek pengembangan sistem admisi Universitas islam Indonesia, yaitu UIIAdmisi.

3.2.1 Program Induksi SPK Baru dan Magang Kerja

Program induksi SPK baru dan magang kerja dilakukan pada tiga hari pertama pelaksanaan magang dimulai pada tanggal 5 September 2022 hingga 7 September 2022.

Program ini adalah proses penyambutan SPK baru dan Magang kerja ke dalam Badan Sistem Informasi UII. Selain itu juga dijelaskan tentang teknologi-teknologi yang digunakan dan bagaimana budaya kerja di Badan Sistem Informasi UII.

Pada hari pertama program induksi dijelaskan tentang nilai sosial *ke-UII-an* di lingkungan kerja Badan Sistem Informasi UII dan juga bagaimana etika kerja di kantor, seperti jam kerja di kantor, bagaimana etika berpakaian di kantor, dan bagaimana etika menghadapi karyawan lain. Mengingat Badan Sistem Informasi UII merupakan badan yang termasuk dalam struktur organisasi Universitas Islam Indonesia, nilai sosial yang dijunjung tidak jauh dari nilai-nilai sosial yang berasal dari hadis maupun Al-Quran.

Pada hari kedua dijelaskan materi yang bertema *working culture*, mulai dari apa saja filosofi dan pilar di dalam *scrum* yang merupakan metode siklus pengembangan perangkat lunak yang digunakan oleh semua tim pengembang yang ada di BSI UII serta bagaimana penerapan di dalam lingkungan kerja lalu *tools* yang digunakan untuk mendukung pelaksanaan *scrum,* yaitu Jira dan Confluence. Yang terakhir dijelaskan bahwa budaya kerja yang ada di BSI UII memegang erat nilai-nilai Islami.

Pada hari ketiga dijelaskan materi yang bertema *development environment* tentang teknologi-teknologi apa saja yang sedang digunakan oleh BSI UII, mulai dari infrastruktur teknologi hingga bagaimana implementasi sistem *microservices* yang sudah digunakan oleh BSI UII. Dijelaskan juga bagaimana arsitektur jaringan di Universitas Islam Indonesia yang dulu hanya bisa menampung *bandwidth* yang kecil bila dibandingkan dengan *bandwidth* yang sekarang berkat perubahan arsitektur jaringan dan bagaimana proses yang terjadi. Serta dijelaskan juga materi tentang *data security* dan kasus-kasus terkait yang pernah terjadi di BSI UII. Tabel 3.1 merupakan jadwal secara rinci dari program induksi SPK baru dan magang kerja yang sudah dilaksanakan.

Tabel 3.1 Jadwal kegiatan program induksi SPK baru dan magang kerja

Hari pertama **Topik: Ke-UII-an**		
No.	**Materi**	**Pemateri**
1	Opening	Kholid Haryono, S.T., M.Kom.
2	BSI product knowledge (support) dan layanan prima	Endro Mustofa, S.Kom.

3	Regulation and ethics	Fitria Prihatini, S.T.
4	Ke-UII-an	Dr. Mukhammad Andri Setiawan
Hari kedua **Topik: Working Culture**		
1	SCRUM Philosopy	Tutik Maryana S.Kom
2	Kanban & Jira - Confluence	Fairuzi Nisrina Rismhandani S.Psi
3	Develop a Champion Culture	Kholid Haryono, S.T., M.Kom.
4	Islamic Culture	Ari Sujarwo, S.Kom, M.I.T.
Hari Ketiga **Topic: Working Culture**		
1	UIIGateway (*Development Architecture & Microservice*)	Andi Purwanto, S.T.
2	Database & Interoperability (*Infra Dev*)	Pandu Bangun Asmoro, S.Kom.
3	NDC & OPS (Security)	Akhmad Rafiuddin, S.Kom. & Frendi Yusroni Romadhona, S.Kom.
4	Closing	Kholid Haryono, S.T., M.Kom.

3.2.2 Pengembangan UIIAdmisi

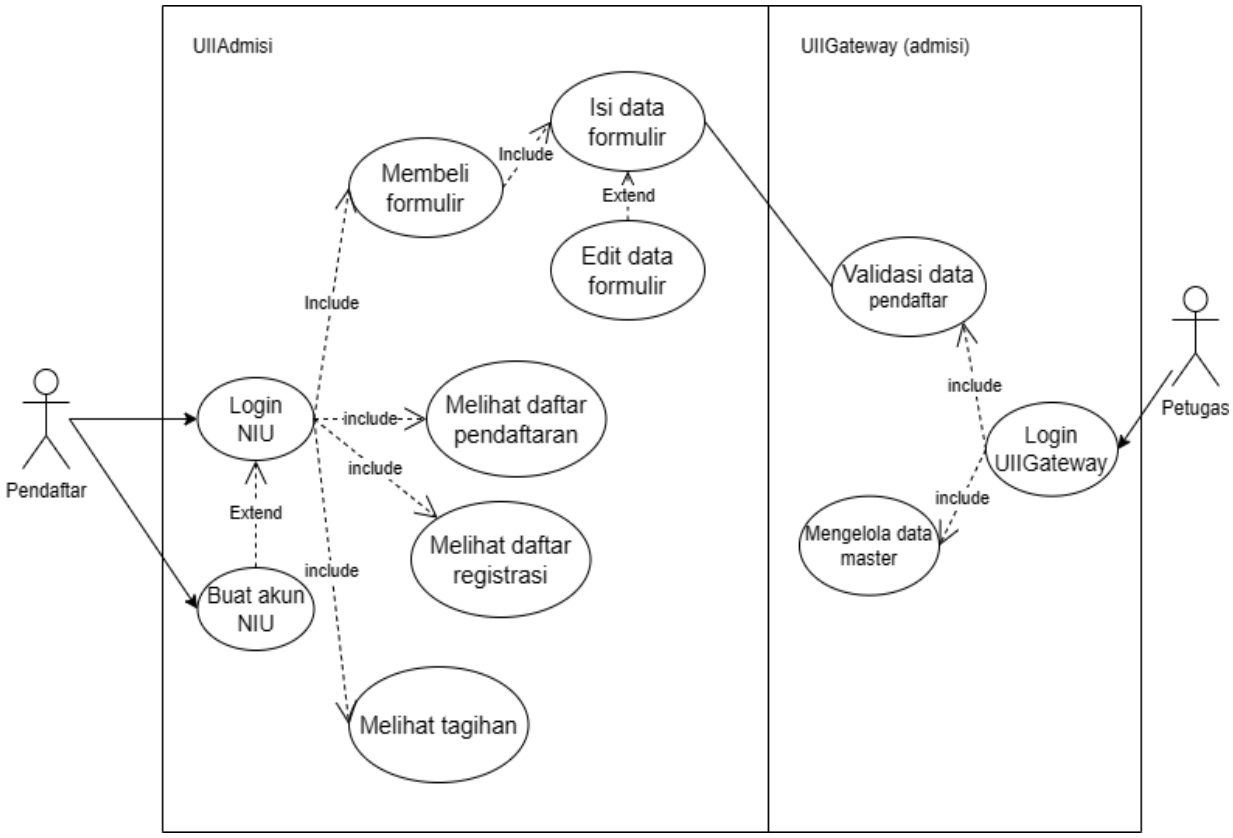
Dalam mengembangkan UIIAdmisi, setiap fitur baru yang dikembangkan akan melalui tahapan *environment* yang terisolasi. Tahapan yang dimaksud adalah tahapan *development*, *staging*, *production*. Tahapan *development* merupakan tahapan fitur mulai dikembangkan, setelah itu fitur memasuki tahapan *staging* untuk dilakukan tes pada aspek fungsional dan non-fungsional oleh *UI/UX designer* tim Admisi, *feedback* dari *UI/UX designer* akan menentukan apakah fitur tersebut akan lanjut ke tahap *production* untuk digunakan oleh *end-user* atau kembali ke tahap *development* untuk dikembangkan lagi. Masing-masing tahapan tersebut memiliki basis data serta hak akses yang berbeda. Isolasi tersebut dilakukan agar proses pengembangan fitur baru dapat berjalan dan tidak mengganggu fitur yang sudah bisa digunakan oleh *end-user* pada saat itu.

Strategic design

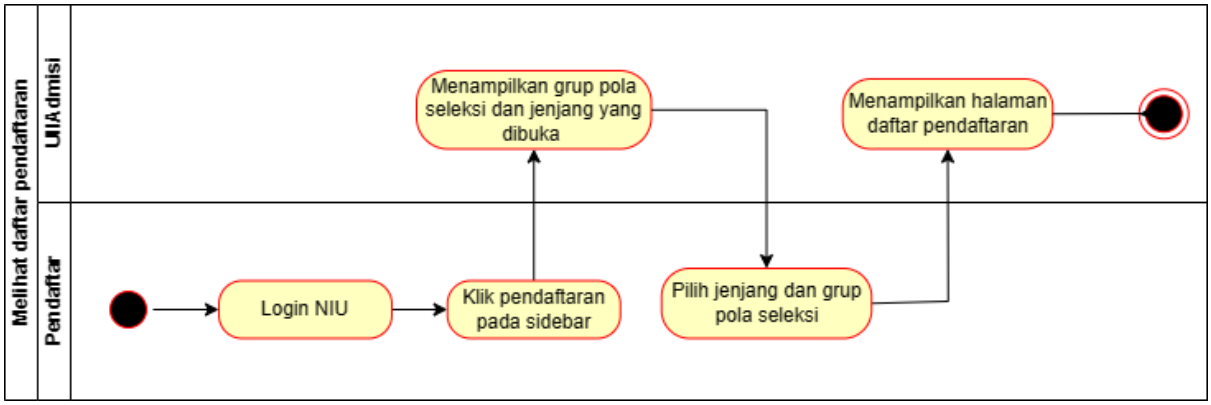
Dalam mengembangkan suatu sistem, tahap paling awal yang selalu dilakukan adalah melakukan analisis kebutuhan pada sistem. Di dalam *scrum*, kebutuhan tersebut akan menjadi *product backlog* yang nanti akan dikerjakan saat iterasi *sprint*. Kebutuhan tersebut ditentukan oleh *UI/UX designer* dan *product owner* melalui proses diskusi dengan para *stakeholder*. Setelah menentukan kebutuhan, *UI/UX designer* akan mulai membuat desain *user interface* sistem. Tabel 3.2 merupakan daftar *product backlog* yang dikerjakan selama magang. *Product backlog* pada Tabel 3.2 akan dijadikan menjadi lebih modular ketika *product backlog* tersebut akan dijadikan *sprint backlog*.

Tujuan utama dalam *strategic design* adalah menghasilkan sebuah mapping dari beberapa *domain* dan *subdomain*, dari mapping tersebut batasan dari masing-masing *domain* akan terlihat dan menjadi acuan dalam menentukan kandidat *microservices*. Proses *strategic design* ini akan dilakukan pada awal iterasi *sprint* di saat tim *scrum* masih menentukan kebutuhan sistem. Langkah pertama yang dilakukan pada *strategic design* adalah menentukan *domain* dan *subdomain* dari permasalahan bisnis yang akan diselesaikan. Salah satu cara untuk menentukan *domain* pada sistem adalah dengan melakukan analisis pada proses bisnis, hasil dari analisis proses bisnis tersebut dapat berupa sebuah *Activity Diagram*.

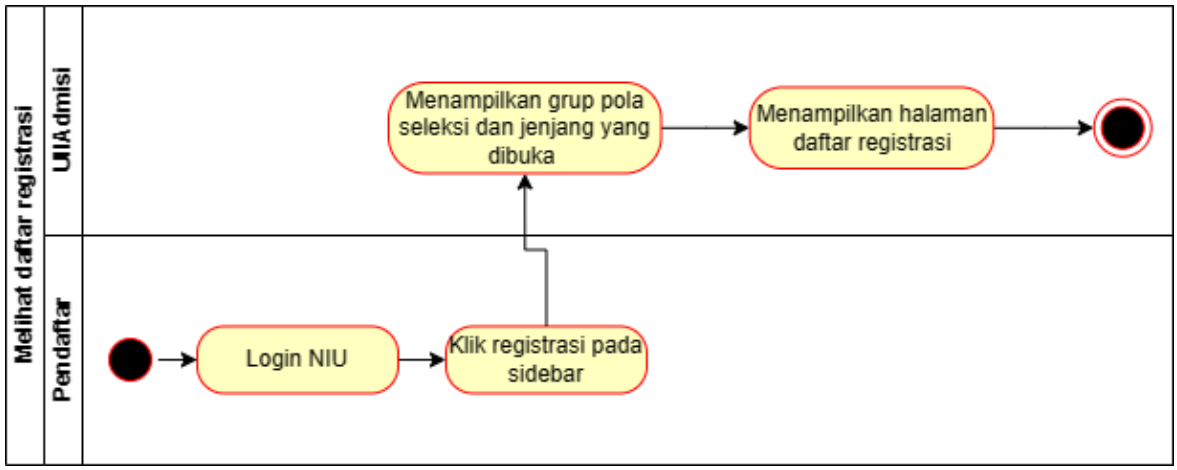
UIIAdmisi memiliki beberapa jenis pengguna yang meliputi calon pendaftar, petugas PMB UII, petugas mitra dan kepala program studi. Dalam mengembangkan UIIAdmisi, kegiatan magang yang dilakukan adalah mengembangkan fitur-fitur yang ada pada sisi calon pendaftar. Gambar 3.9 adalah *Use Case Diagram* (UCD) dari UIIAdmisi pada sisi pendaftar dan Gambar 3.10 hingga Gambar 3.15 adalah *Activity Diagram* dari beberapa *use case* pada Gambar 3.9. UCD dapat memberikan gambaran skenario-skenario yang melibatkan pengguna dengan sistem.



Gambar 3.9 UCD sistem UIIAdmisi sisi mahasiswa

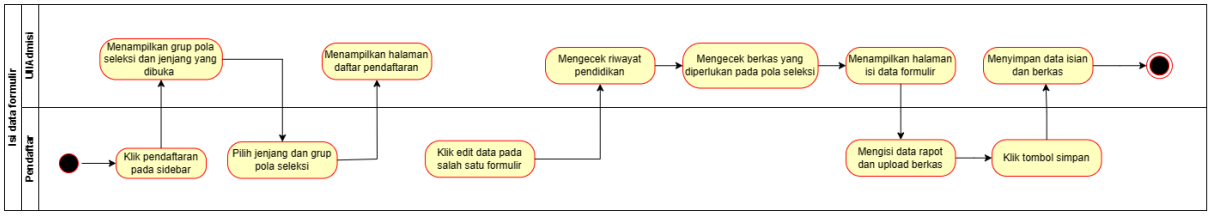


Gambar 3.10 Activity Diagram melihat daftar formulir pendaftaran

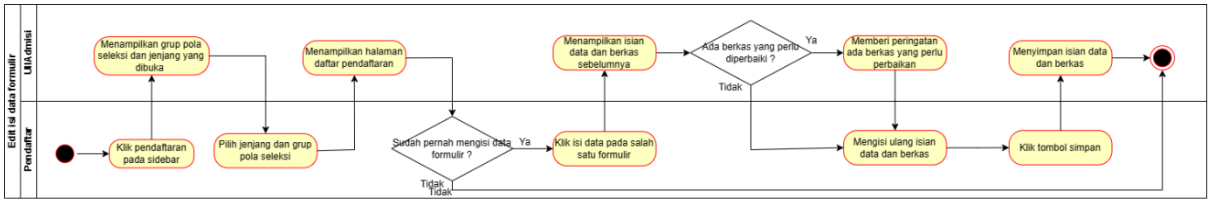


Gambar 3.11 Activity Diagram melihat daftar registrasi

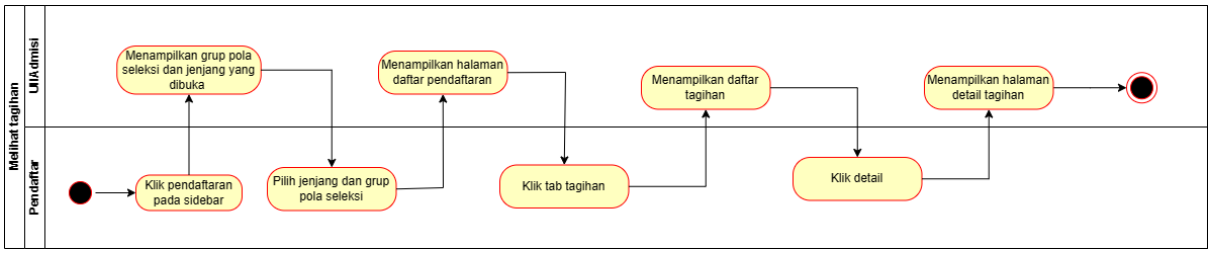
Gambar 3.12 Activity Diagram beli formulir



Gambar 3.13 Activity Diagram isi data formulir



Gambar 3.14 Activity Diagram edit data formulir



Gambar 3.15 Activity Diagram melihat tagihan

Tabel 3.2 *Product backlog* yang dikerjakan selama magang

No.	Product Backlog	Deskripsi
1	Beli formulir	Pendaftar dapat melakukan proses membeli formulir pada grup pola seleksi SIBER, CBT, PSB, PBT dengan kebijakan yang berlaku sesuai di pola seleksi.
2	Isi data formulir	Pendaftar dapat melakukan proses mengisi data formulir pada grup pola seleksi SIBER, CBT, PSB, PBT. Dan juga pendaftar dapat melakukan edit isian data ketika petugas mengharuskan pendaftar untuk melakukan perbaikan isian data.
3	Tagihan	Pendaftar dapat melihat daftar tagihan yang terbuat ketika pendaftar melakukan pembelian formulir pendaftaran dan ketika pendaftar telah diterima di salah satu program studi. Pendaftar juga bisa melihat detil tagihan, seperti nomor tagihan, tempat pembayaran tagihan, nominal tagihan, dan batas waktu pembayaran tagihan.
4	*List* pendaftaran	Pendaftar dapat melihat daftar formulir pendaftaran yang diikuti pada masing-masing pola seleksi. Pendaftar juga dapat melihat sudah sampai tahap mana pendaftar berada pada setiap pola seleksi yang diikuti.

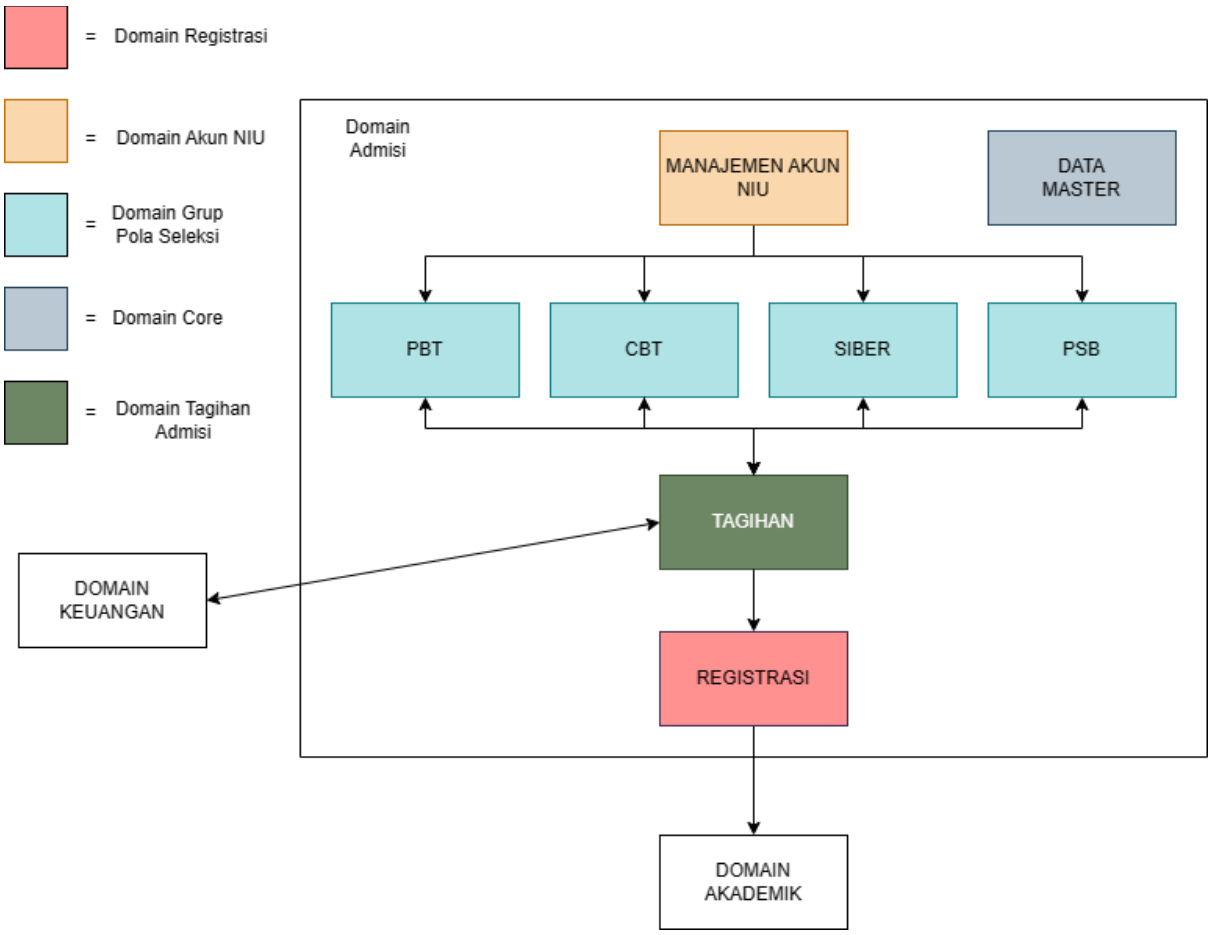
5	Rekomendasi	Pendaftar dapat melihat daftar program studi rekomendasi yang di dapat ketika pendaftar tidak diterima pada semua program studi di suatu formulir. Pendaftar juga dapat memilih program studi rekomendasi dan menjadi diterima di saat memilih rekomendasi tersebut.
6	*List* registrasi	Pendaftar dapat melihat daftar formulir pendaftaran yang telah berstatus diterima. Pendaftar akan melalui proses registrasi yang mana pada akhir proses ini, pendaftar akan memiliki akun pendidikan UII. Pendaftar akan dapat melihat tahap apa yang pendaftar sedang lakukan sekarang.

Setelah melakukan analisis terhadap kebutuhan dan proses bisnis UIIAdmisi didapatkan beberapa *domain* yang ada pada domain bisnis UIIAdmisi, yaitu:

a. Grup pola seleksi.
b. Akun NIU.
c. Tagihan.
d. Registrasi.
e. Core.

Setelah menentukan *domain* dan *subdomain*, langkah berikutnya adalah menentukan relasi antar *subdomain,* gambar 3.16 merupakan hasil dari gambaran relasi dari masing-masing *subdomain*. Berdasarkan gambar 3.16 akan didapatkan beberapa kandidat *microservice,* yaitu *microservice* pendaftaran NIU, *microservice* SIBER, *microservice* PBT, *microservice* CBT dan *microservice* PSB, *microservice* tagihan, *microservice* registrasi, dan *microservice* data *master*. *Microservice* pendaftaran NIU akan digunakan untuk melayani proses pendaftaran akun NIU dan juga pengambilan data akun NIU untuk dapat ditampilkan di klien. *Microservices* SIBER, PBT, CBT dan PSB akan digunakan untuk melayani proses pendaftaran pada masing-masing grup pola seleksi, hal ini dilakukan karena pada masing-masing grup pola seleksi memiliki jenis pola seleksi yang memiliki kebijakan dan aturan bisnis yang berbeda. *Microservice* registrasi akan digunakan untuk melayani proses registrasi pendaftar yang telah

diterima, seperti unggah berkas registrasi dan integrasi ke sistem pada bidang bisnis akademik hingga pendaftar mendapatkan Nomor Induk Mahasiswa (NIM). *Microservice* tagihan akan digunakan untuk melayani proses pembuatan tagihan dan menangani integrasi ke sistem pada bidang bisnis keuangan. *Microservice* data *master* akan digunakan sebagai *microservice* yang membantu *microservice* lain dengan menyediakan data yang lengkap ketika *microservice* lain membutuhkan.

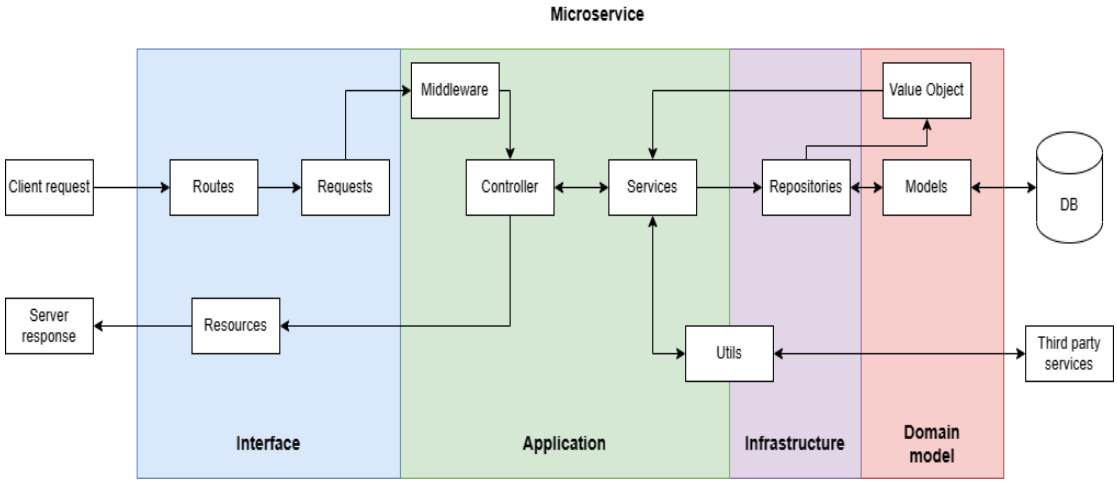


Gambar 3.16 Peta calon komponen sistem UIIAdmisi calon pendaftar

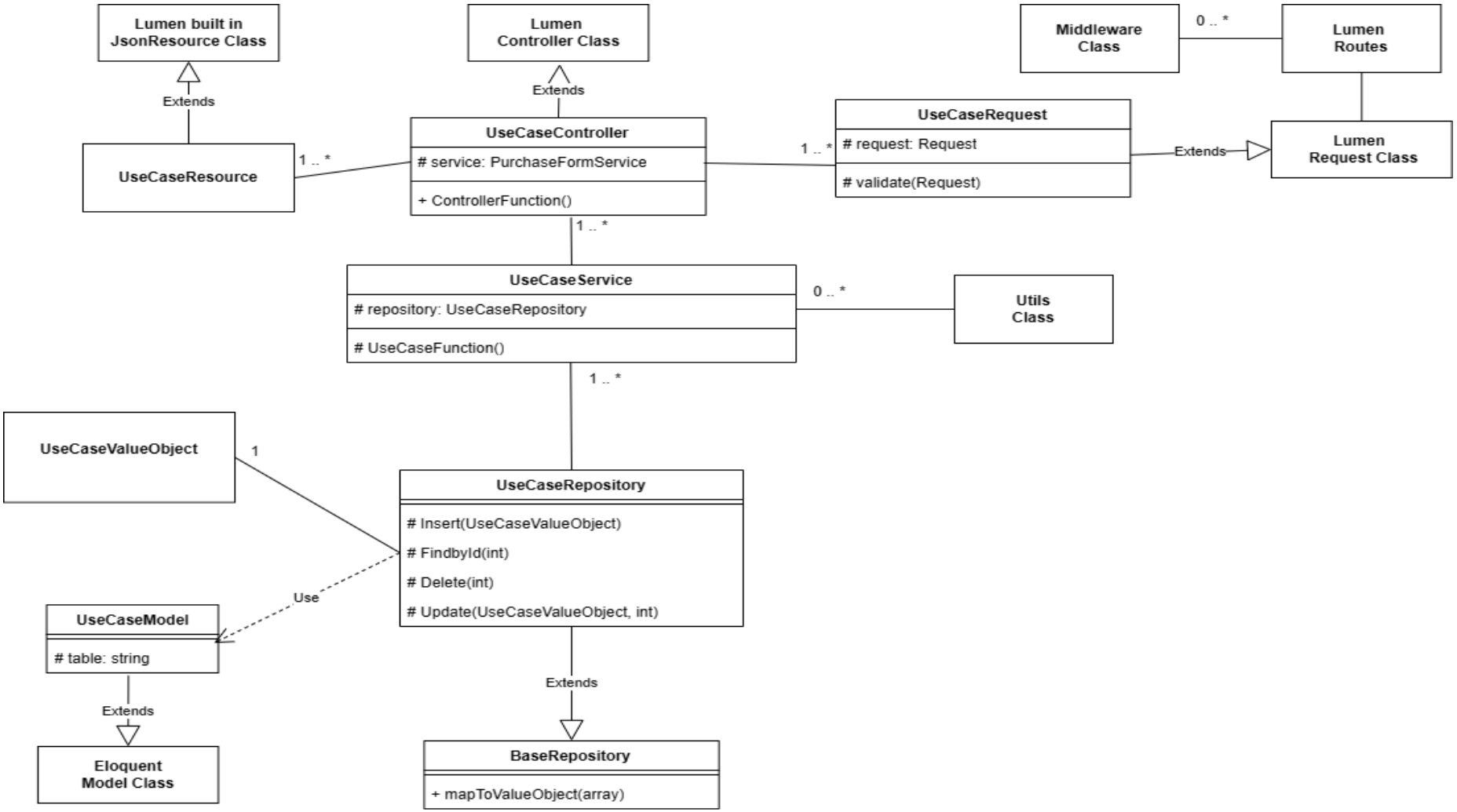
Tactical Design

Tactical design bertujuan untuk mengimplementasikan model yang telah dibuat pada *strategic design* menjadi kode dengan menggunakan *building blocks* yang ada pada *tactical design*. Selain itu, RESTful API diterapkan untuk memungkinkan komunikasi antara *microservice* dan *front-end*. Dalam sistem UIIAdmisi *layer* yang ada pada *clean architecture* diterapkan pada masing-masing *microservice*. Gambar 3.17 merupakan gambaran implementasi *clean architecture* dan *building blocks tactical design* pada masing-masing

microservice. Framework Laravel Lumen juga akan digunakan pada masing-masing *microservice* untuk mempermudah proses pengembangan UIIAdmisi dengan menerapkan fitur-fitur yang ada pada Laravel Lumen. Penggunaan fitur-fitur pada Laravel Lumen juga akan mempermudah implementasi *building blocks* yang ada di *tactical design.* Gambar 3.18 adalah *Class Diagram* dari *microservice* Laravel Lumen untuk masing-masing *use case* yang telah mengimplementasikan *tactical design* dan *clean architecture*.



Gambar 3.17 Implementasi *clean architecture* dan *tactical design* pada *microservice* Laravel Lumen



Gambar 3.18 *Class Diagram* dari *microservice* Laravel Lumen untuk setiap *use case*

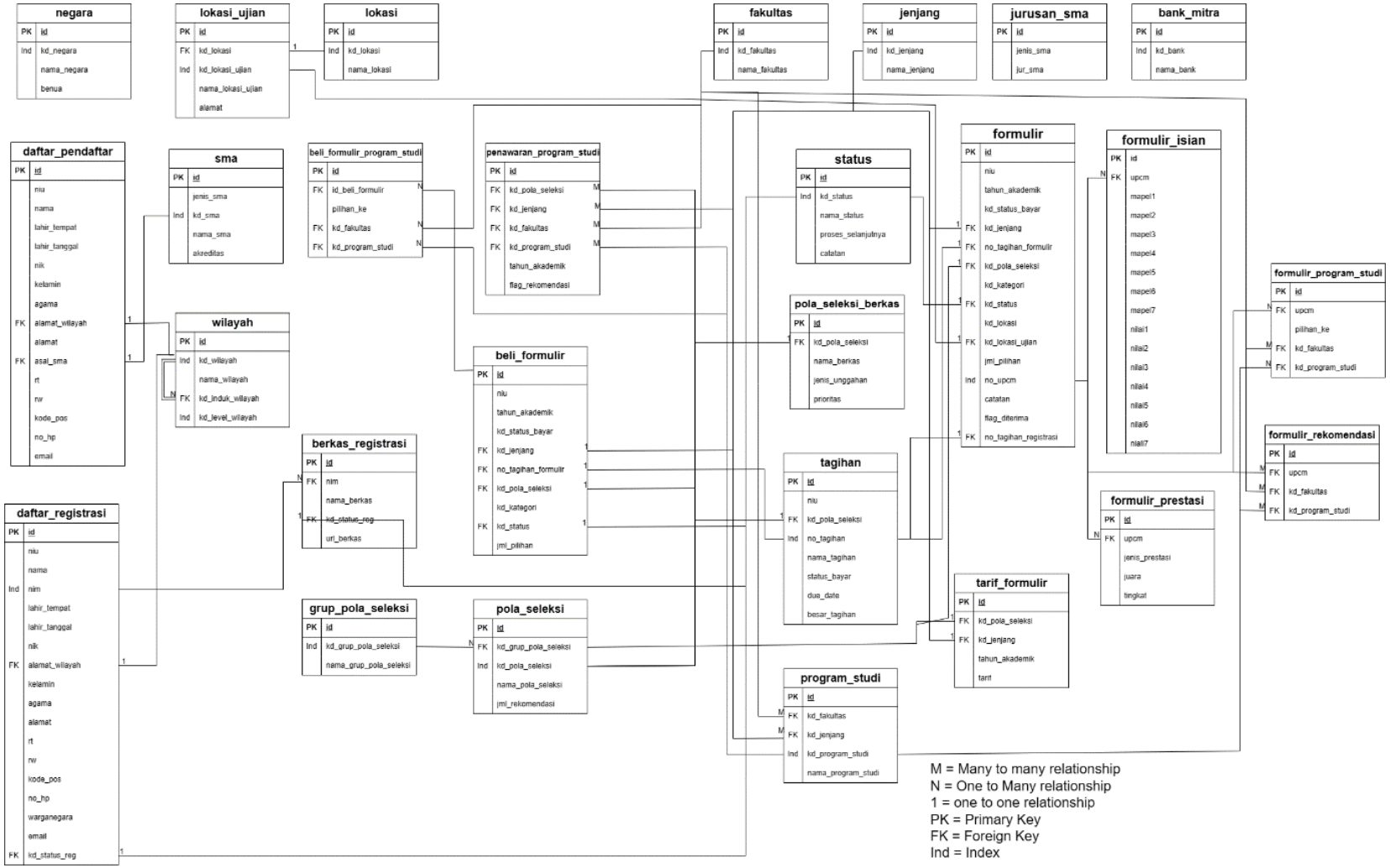
Komponen pada gambar 3.17 akan dibuat menjadi sebuah *module* yang di dalamnya terdapat sebuah *class* yang mendukung sebuah *use case*. Penjelasan dari beberapa *module* pada *microservice,* yaitu:

a. *Controller,* pada *module controller* akan terdapat beberapa *class controller* yang akan digunakan untuk mengontrol *request* dari klien.
b. *Middleware,* pada *module Middleware* akan terdapat beberapa *class* yang bertanggung jawab untuk men-*filter request* dari klien. Di dalam *module middleware* terdapat *class auth* yang mengecek autentikasi klien ketika *request* dan class CORS *middleware* yang mengatur perizinan mekanisme *Cross Origin Resource Sharing* (CORS).
c. *Requests,* pada *module requests* terdapat beberapa *class requests* yang akan melakukan validasi terhadap isi *request body* atau *request query* setiap klien melakukan *request* ke server.
d. *Resource,* pada *module* ini akan ada beberapa *class resource* merupakan komponen yang akan men-*filter* informasi agar sesuai dengan yang dibutuhkan oleh klien dan mengubah informasi tersebut menjadi bentuk JSON.
e. *Model,* pada *module model* akan terdapat beberapa *class model* yang menggunakan fitur Eloquent ORM pada Laravel yang memungkinkan untuk menghubungkan suatu *class model* ke suatu tabel pada basis data.
f. *Repositories,* pada *module repositories* akan terdapat beberapa *class repository* yang fungsi utamanya untuk melakukan proses modifikasi pada basis data dengan *query.*
g. *Services,* pada *module services* akan terdapat beberapa *class service* yang berisikan logika bisnis suatu *domain model*. *Class* ini akan terhubung dengan *class repositories* untuk dapat mengakses *entities* yang nanti akan diolah pada *class* ini.
h. *Routes,* pada *module routes* akan terdapat sebuah *class route* yang berfungsi sebagai *interface* yang memetakan rute untuk menuju suatu *resource*.
i. *Utils,* pada *module* ini akan terdapat beberapa *class* yang menyediakan fungsi pembantu, seperti sebuah *class* yang membantu proses mengunggah sebuah dokumen ke layanan ketiga.

Setelah mengimplementasikan building blocks dan clean architecture untuk *microservice* Laravel Lumen, proses berikutnya yang dilakukan adalah mendesain basis data. Pada pengembangan UIIAdmisi, basis data yang digunakan adalah basis data relasional MySQL. Pada MySQL terdapat tabel yang menjadi struktur dasar menyimpan data yang disusun dalam bentuk baris dan kolom, dimana setiap tabel akan mewakili suatu tipe entitas. Data yang

terdapat pada MySQL bersifat terstruktur dan memiliki skema yang ketat. MySQL juga dapat melakukan penggabungan data dari beberapa tabel dengan melakukan *join*. Sebelum melakukan *join*, terlebih dahulu harus didefinisikan hubungan antara tabel menggunakan *foreign key* dengan kardinalitas relasi, yaitu *one-to-one*, *one-to-many*, *many-to-many*. Jika dibandingkan dengan basis data non-relasional, MySQL memiliki performa yang lebih baik dan konsisten dikarenakan skema data yang terstruktur dan ketat yang mana cocok dengan UIIAdmisi karena memiliki struktur data yang tidak berubah.

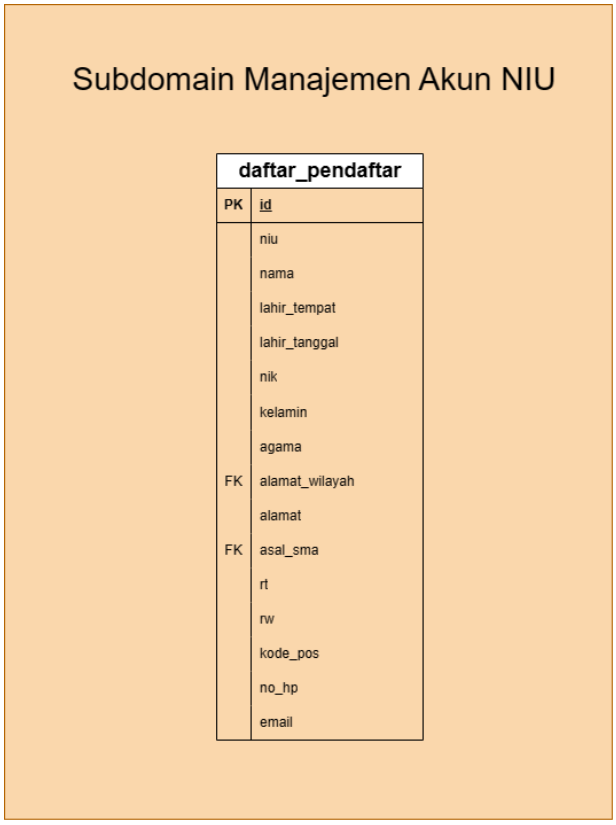
Untuk mendesain basis data UIIAdmisi, Entity Relationship Diagram (ERD) digunakan untuk menggambarkan struktur pada basis data. Penggunaan ERD ini dapat membantu dalam pemahaman yang lebih baik tentang basis data yang didesain. Selain itu, penggunaan ERD juga dapat menggambarkan model data tingkat tinggi yang mendefinisikan tabel, atribut pada kolom, *index* pada tabel, serta hubungan antar tabel. Gambar 3.19 merupakan *Entity Relationship Diagram* (ERD) dari basis data sistem UIIAdmisi. Tabel pada basis data tersebut akan dikelompokkan dan dimasukkan ke dalam basis data masing-masing *microservice.* Beberapa tabel akan ditempatkan di lebih dari satu *microservice*, seperti tabel formulir yang akan ditempatkan di semua *microservice* grup pola seleksi. Setiap tabel akan dikonfigurasi untuk memiliki *primary key* pada kolom id untuk dapat memastikan bahwa setiap baris data pada tabel memiliki identifikasi yang unik yang sangat penting untuk menghindari duplikasi data.



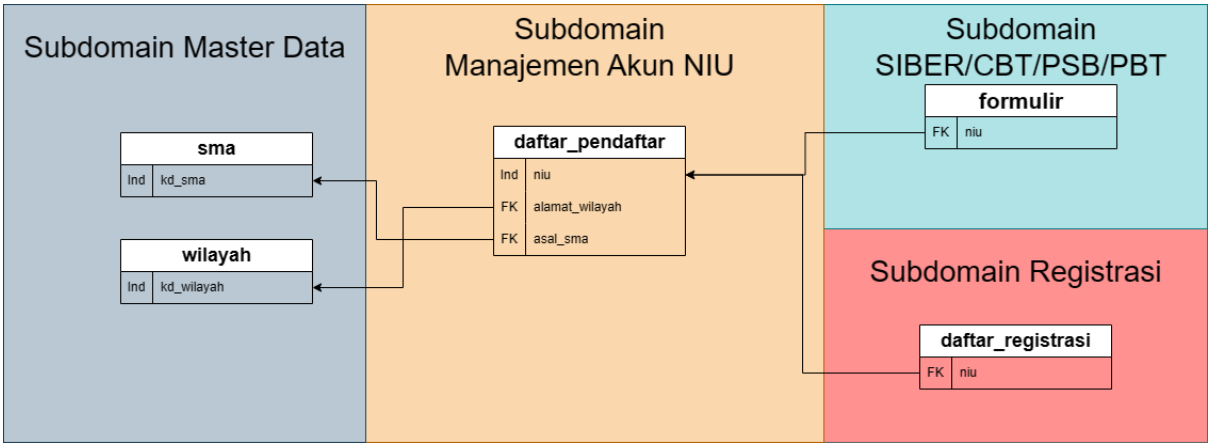
Gambar 3.19 ERD dari basis data UIIAdmisi

a. *Microservice* manajemen NIU

Pada *microservice* ini hanya terdapat tabel daftar pendaftar struktur tabel dapat dilihat pada Tabel 3.3. Gambar 3.20 adalah ERD *subdomain* manajemen akun NIU dan Gambar 3.21 adalah pemetaan hubungan tabel *subdomain* manajemen akun NIU dengan *subdomain* lain.



Gambar 3.20 ERD dari *microservice* manajemen NIU



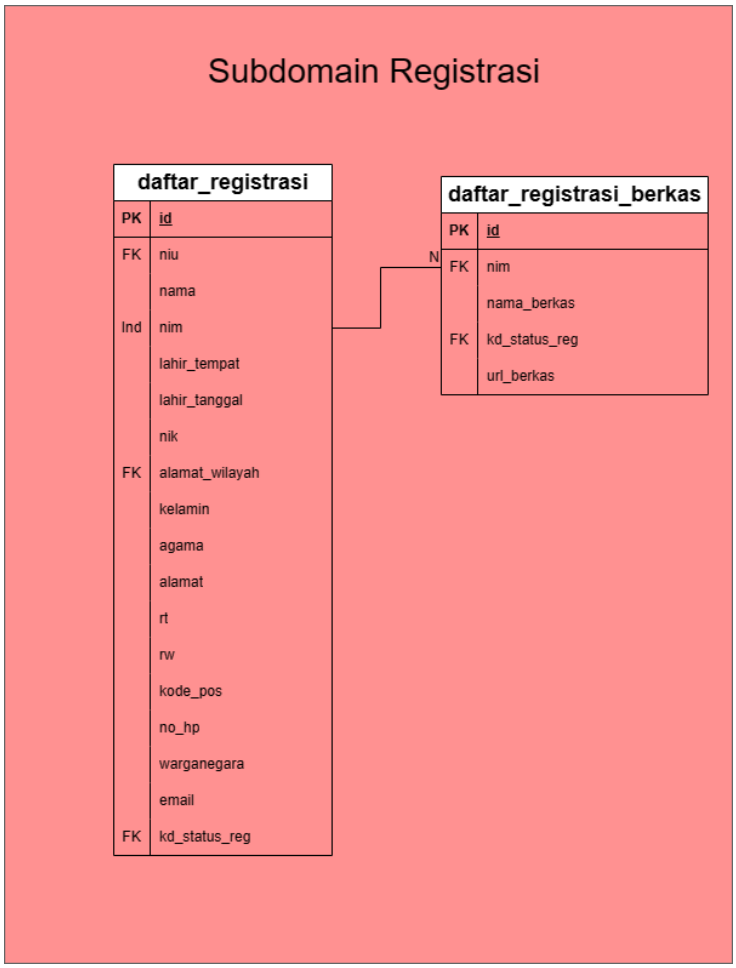
Gambar 3.21 Mapping hubungan tabel *subdomain* manajemen akun NIU

Tabel 3.3 Struktur tabel daftar_pendaftar

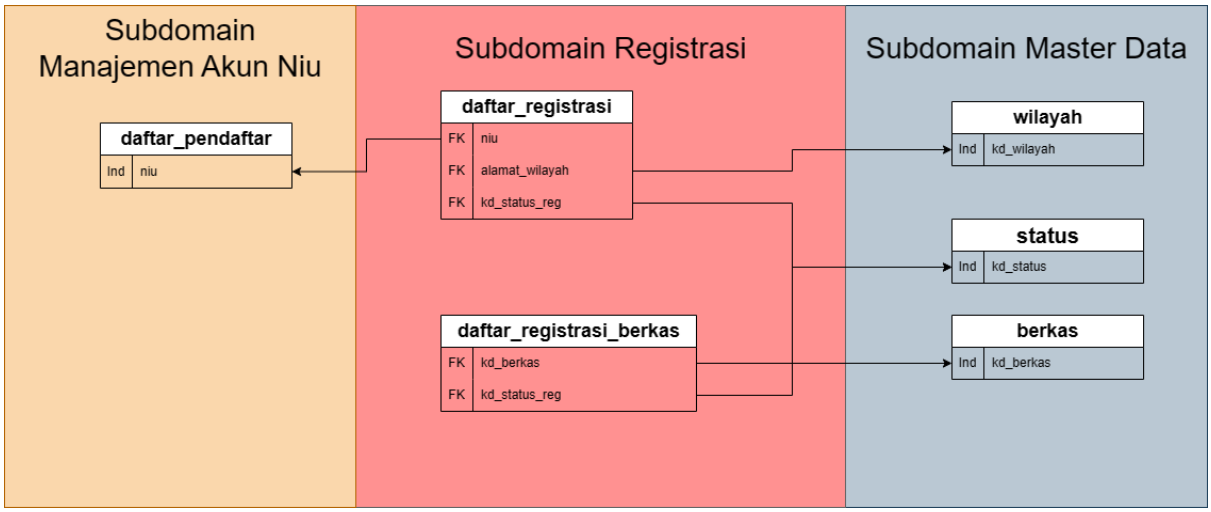
Kolom	Tipe data
id	int
niu	varchar
nama	varchar
lahir_tempat	varchar
lahir_tanggal	date
nik	int
id_kelamin	int
id_agama	int
alamat	text
rt	int
rw	int
kode_pos	int

b. *Microservice* registrasi

Pada *microservice* registrasi terdapat tabel daftar_registrasi dan berkas_registrasi. Struktur tabel dapat dilihat dari Tabel 3.4 dan Tabel 3.5. Gambar 3.22 adalah ERD *subdomain* registrasi dan Gambar 3.23 adalah pemetaan hubungan tabel *subdomain* registrasi dengan *subdomain* lain.



Gambar 3.22 ERD dari *microservice* registrasi



Gambar 3.23 Mapping hubungan tabel *subdomain* registrasi

Tabel 3.4 Struktur tabel daftar_registrasi

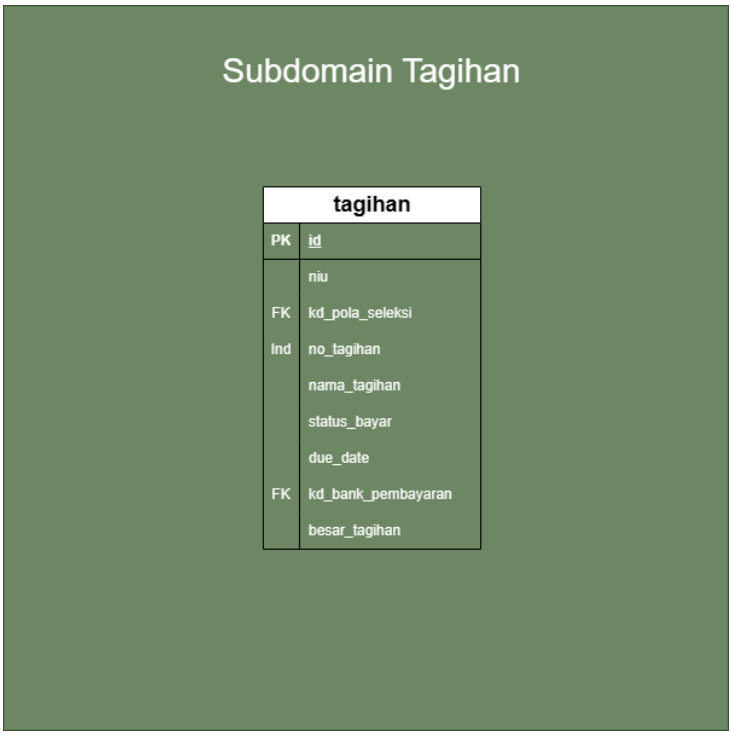
Kolom	Tipe data
id	int
niu	varchar
nama	varchar
lahir_tempat	varchar
lahir_tanggal	date
nik	int
id_kelamin	int
id_agama	int
alamat	text
rt	int
rw	int
kode_pos	int
no_hp	int
kd_warganegara	int
email	varchar
kd_status_reg	int

Tabel 3.5 Struktur tabel berkas_registrasi

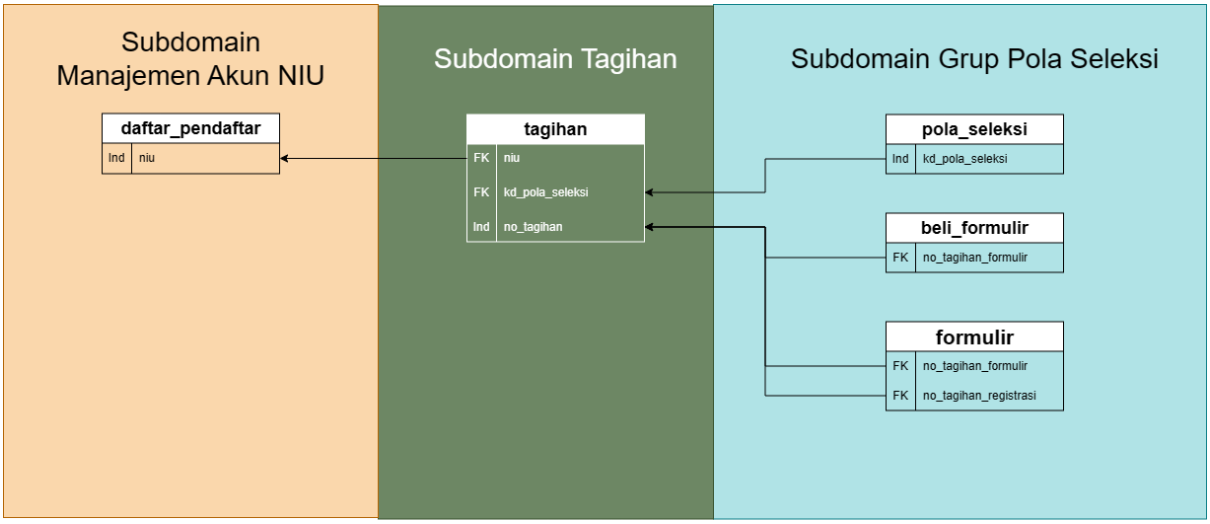
Kolom	Tipe data
id	int
id_master_berkas	int
nim	varchar
kd_step_reg	int
url_berkas	varchar

c. *Microservice* tagihan

Pada *microservice* tagihan hanya terdapat tabel daftar_tagihan dengan struktur tabel yang dapat dilihat pada Tabel 3.6. Gambar 3.24 adalah ERD *subdomain* tagihan dan Gambar 3.25 adalah pemetaan hubungan tabel *subdomain* tagihan dengan *subdomain* lain.



Gambar 3.24 ERD dari *microservice* tagihan



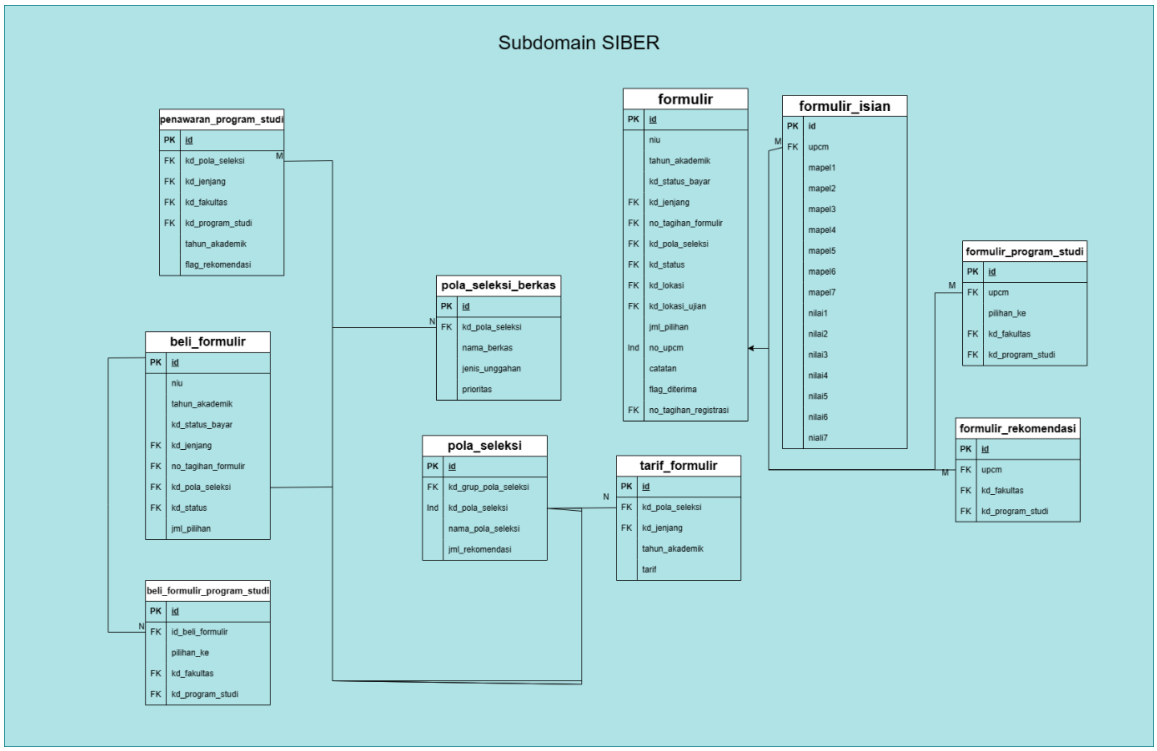
Gambar 3.25 Mapping hubungan tabel *subdomain* tagihan

Tabel 3.6 Struktur tabel daftar tagihan

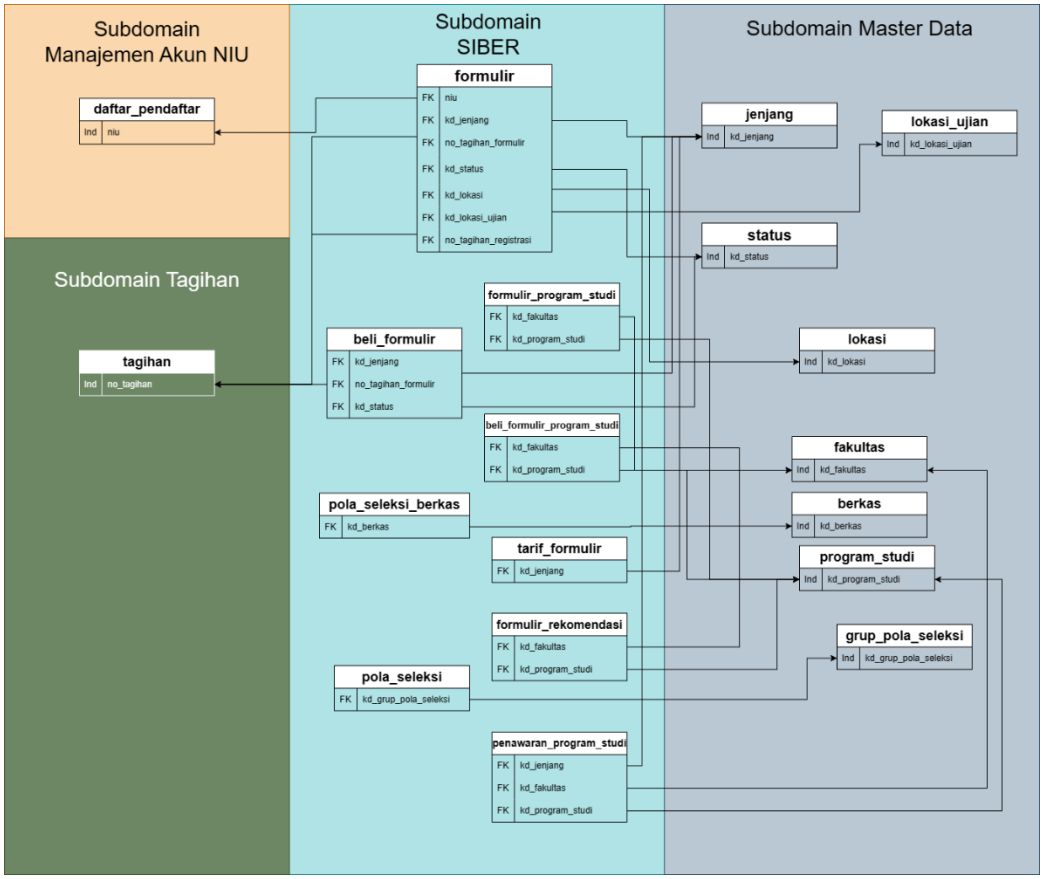
Kolom	Tipe data
id	int
niu	varchar
kd_pola_seleksi	int
no_tagihan	int
nama_tagihan	varchar
status_bayar	int
due_date	date
besar_tagihan	int

d. *Microservice* SIBER

Pada *microservice* SIBER terdapat tabel beli_formulir, beli_formulir_program_studi, formulir (tabel utama), formulir_isian, formulir_program_studi, formulir_rekomendasi, formulir_prestasi, pola_seleksi, pola_seleksi_berkas, tarif_formulir, penawaran_program_studi. Struktur tabel dapat dilihat dari tabel 3.7 hingga tabel 3.17. Gambar 3.26 adalah ERD *subdomain SIBER* dan Gambar 3.27 adalah pemetaan hubungan tabel *subdomain SIBER* dengan *subdomain* lain.



Gambar 3.26 ERD dari *microservice* SIBER



Gambar 3.27 Mapping hubungan tabel *subdomain* SIBER

Tabel 3.7 Struktur tabel beli_formulir

Kolom	Tipe data
id	int
niu	varchar
tahun_akademik	varchar
kd_status_bayar	varchar
no_tagihan_formulir	varchar
kd_jenjang	varchar
kd_pola_seleksi	varchar
kd_kategori	varchar
kd_status	int
jml_pilihan	int

Tabel 3.8 Struktur tabel beli_formulir_program_studi

Kolom	Tipe data
id	int
id_beli_formulir	int
pilihan_ke	int
kd_fakultas	varchar
kd_program_studi	varchar

Tabel 3.9 Struktur tabel formulir (tabel utama)

Kolom	Tipe data
id	int
niu	int
no_upcm	int
tahun_akademik	varchar

kd_jenjang	varchar
no_tagihan_formulir	int
kd_pola_seleksi	varchar
kd_status	int
jml_pilihan	int
kd_fakultas_diterima	varchar
kd_prodi_diterima	varchar
catatan	varchar
flag_diterima	int
tgl_hasil	date
no_tagihan_registrasi	int

Tabel 3.10 Struktur tabel formulir_isian

Kolom	**Tipe data**
id	int
id_formulir	int
kd_mapel1	varchar
kd_mapel2	varchar
kd_mapel3	varchar
kd_mapel4	varchar
kd_mapel5	varchar
kd_mapel6	varchar
kd_mapel7	varchar
nilai1	int
nilai2	int
nilai3	int
nilai4	int

nilai5	int
nilai6	int
nilai7	int

Tabel 3.11 Struktur tabel formulir_program_studi

Kolom	Tipe data
id	int
id_formulir	int
pilihan_ke	int
kd_fakultas	varchar
kd_program_studi	varchar

Tabel 3.12 Struktur tabel formulir_rekomendasi

Kolom	Tipe data
id	int
id_formulir	int
kd_fakultas	varchar
kd_program_studi	varchar
peringkat	int

Tabel 3.13 Struktur tabel formulir_prestasi

Kolom	Tipe data
id	int
id_formulir	int
id_jenis_prestasi	int
juara	int
tingkat	varchar

Tabel 3.14 Struktur tabel pola_seleksi

Kolom	Tipe data
id	int
kd_grup_pola_seleksi	varchar
kd_pola_seleksi	varchar
nama_pola_seleksi	varchar
jml_rekomendasi	int

Tabel 3.15 Struktur tabel pola_seleksi_berkas

Kolom	Tipe data
id	int
kd_pola_seleksi	varchar
nama_berkas	varchar
jenis_unggahan	enum
prioritas	enum

Tabel 3.16 Struktur tabel tarif_formulir

Kolom	Tipe data
id	int
kd_pola_seleksi	varchar
tahun_akademik	varchar
kd_jenjang	varchar
tarif	int

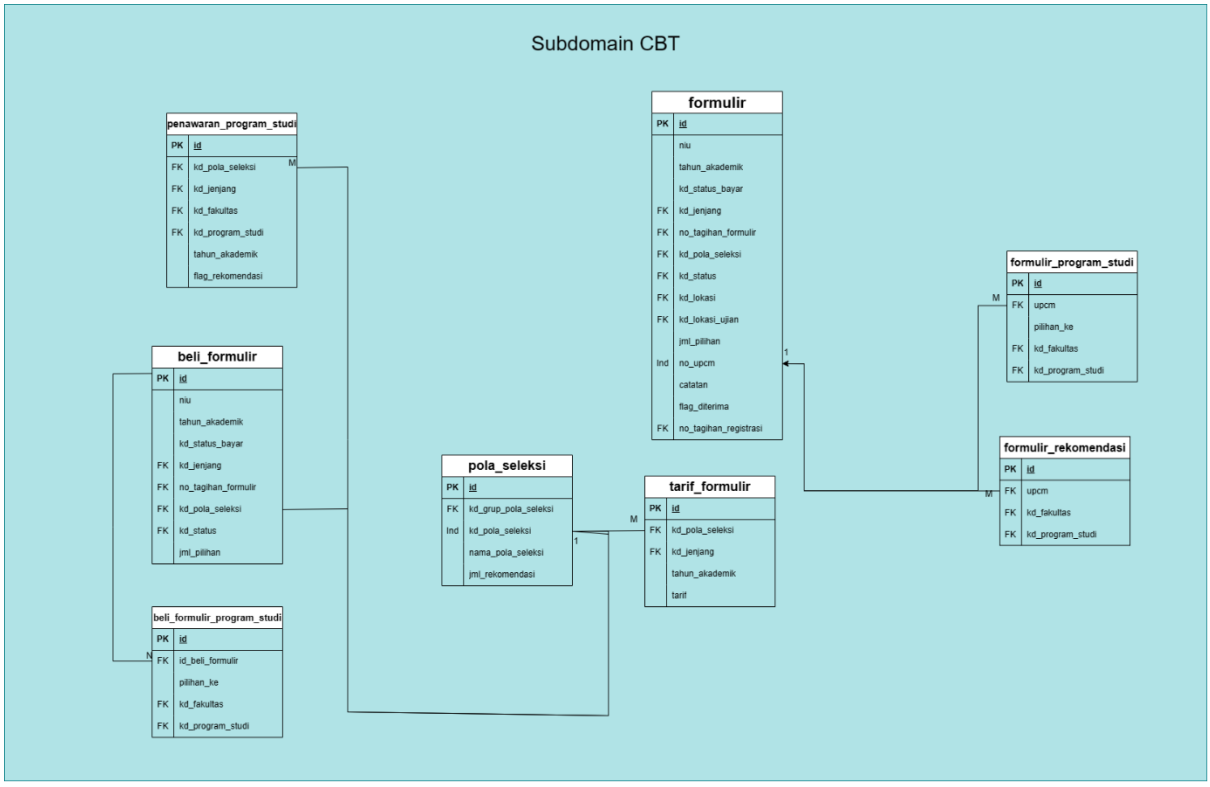
Tabel 3.17 Struktur tabel penawaran_program_studi

Kolom	Tipe data
id	int

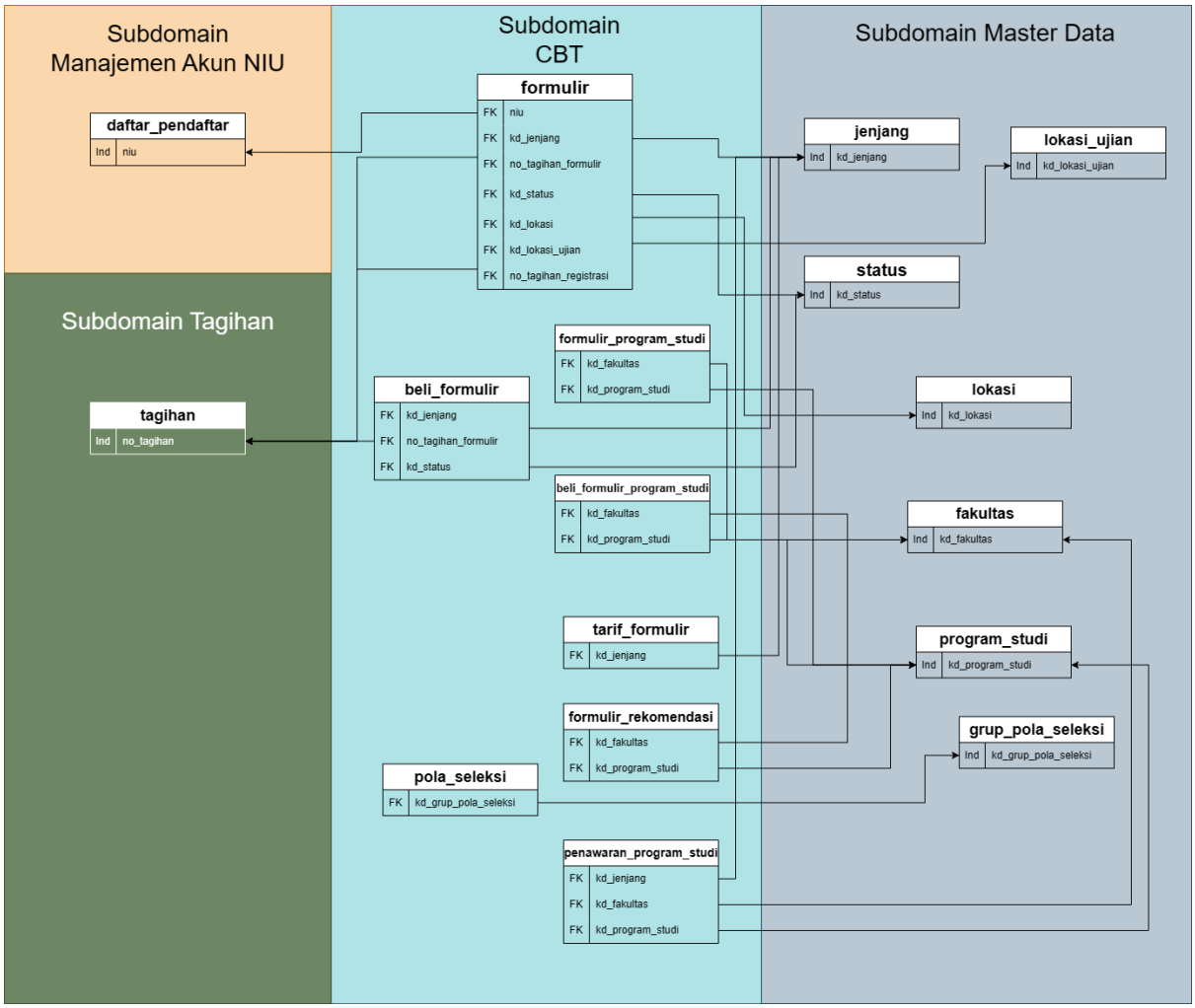
kd_pola_seleksi	varchar
tahun_akademik	varchar
kd_jenjang	varchar
kd_fakultas	varchar
kd_program_studi	varchar
flag_rekomendasi	int

e. *Microservice* CBT

Pada *microservice* CBT terdapat beberapa tabel yang memiliki struktur yang sama dengan tabel *microservice* SIBER, hanya tabel formulir (tabel utama) dan beli_formulir yang memiliki struktur tabel yang berbeda. Tabel 3.18 merupakan struktur tabel formulir dam tabel 3.19 merupakan tabel beli_formulir. Gambar 3.28 adalah ERD subdomain *CBT* dan Gambar 3.29 adalah pemetaan hubungan tabel subdomain *CBT* dengan subdomain lain.



Gambar 3.28 ERD dari *microservice* CBT



Gambar 3.29 Mapping hubungan tabel *subdomain* CBT

Tabel 3.18 Struktur tabel formulir (tabel utama)

Kolom	Tipe data
id	int
niu	int
no_upcm	int
tahun_akademik	varchar
kd_jenjang	varchar
no_tagihan_formulir	int
kd_pola_seleksi	varchar
kd_status	int

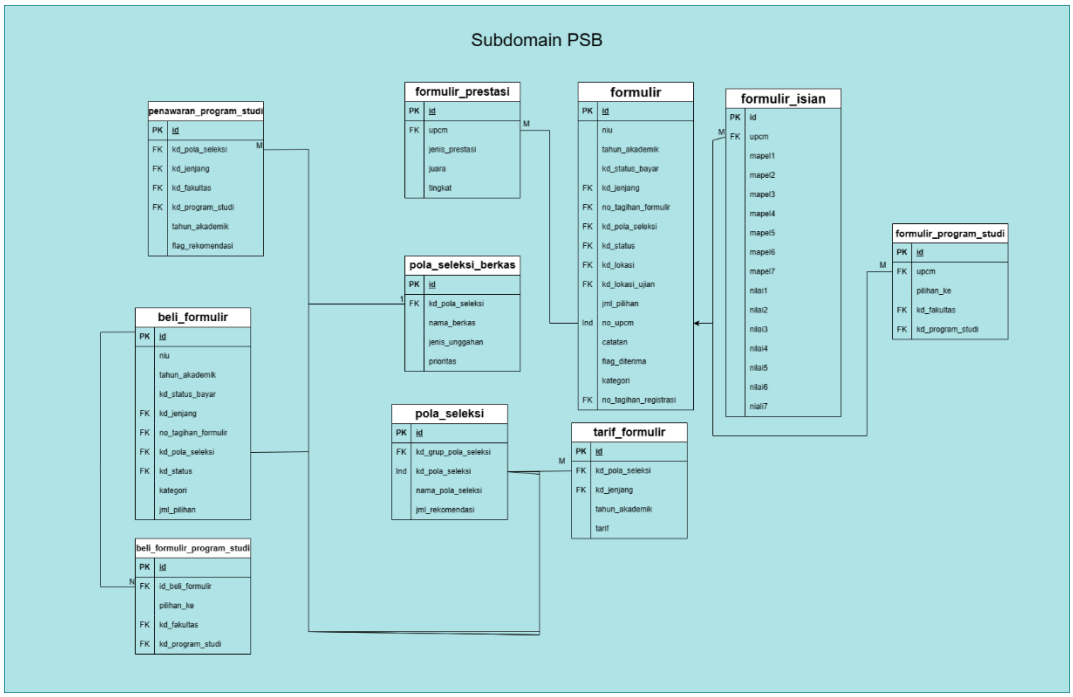
kd_lokasi	int
jml_pilihan	int
kd_fakultas_diterima	varchar
kd_prodi_diterima	varchar

Tabel 3.19 Struktur tabel beli_formulir

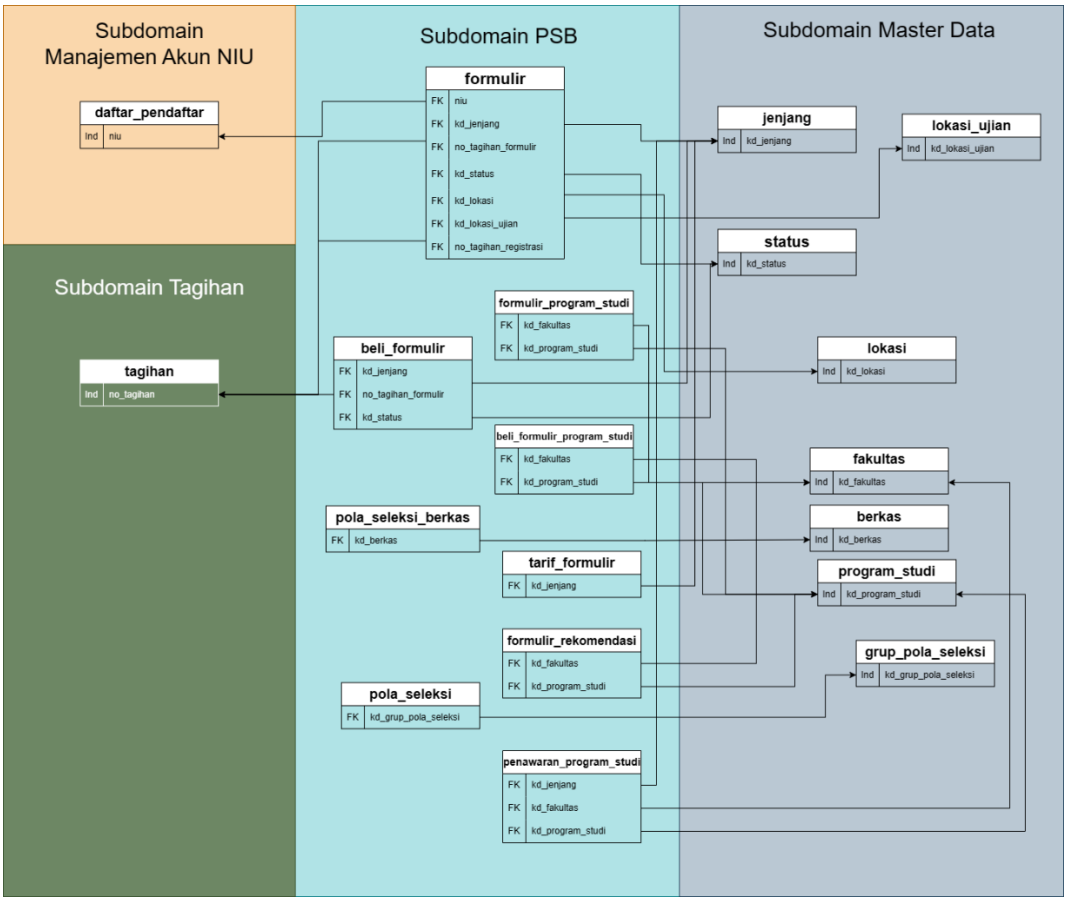
Kolom	Tipe data
id	int
niu	varchar
tahun_akademik	varchar
kd_status_bayar	varchar
no_tagihan_formulir	varchar
kd_jenjang	varchar
kd_pola_seleksi	varchar
kd_kategori	varchar
kd_status	int
jml_pilihan	int
kd_lokasi	int

f. *Microservice* PSB

Microservice PSB memiliki tabel yang sama dengan *microservice* SIBER dan struktur tabel juga sama dengan *microservice* SIBER. Gambar 3.28 adalah ERD subdomain *PSB* dan Gambar 3.29 adalah pemetaan hubungan tabel subdomain *PSB* dengan subdomain lain.



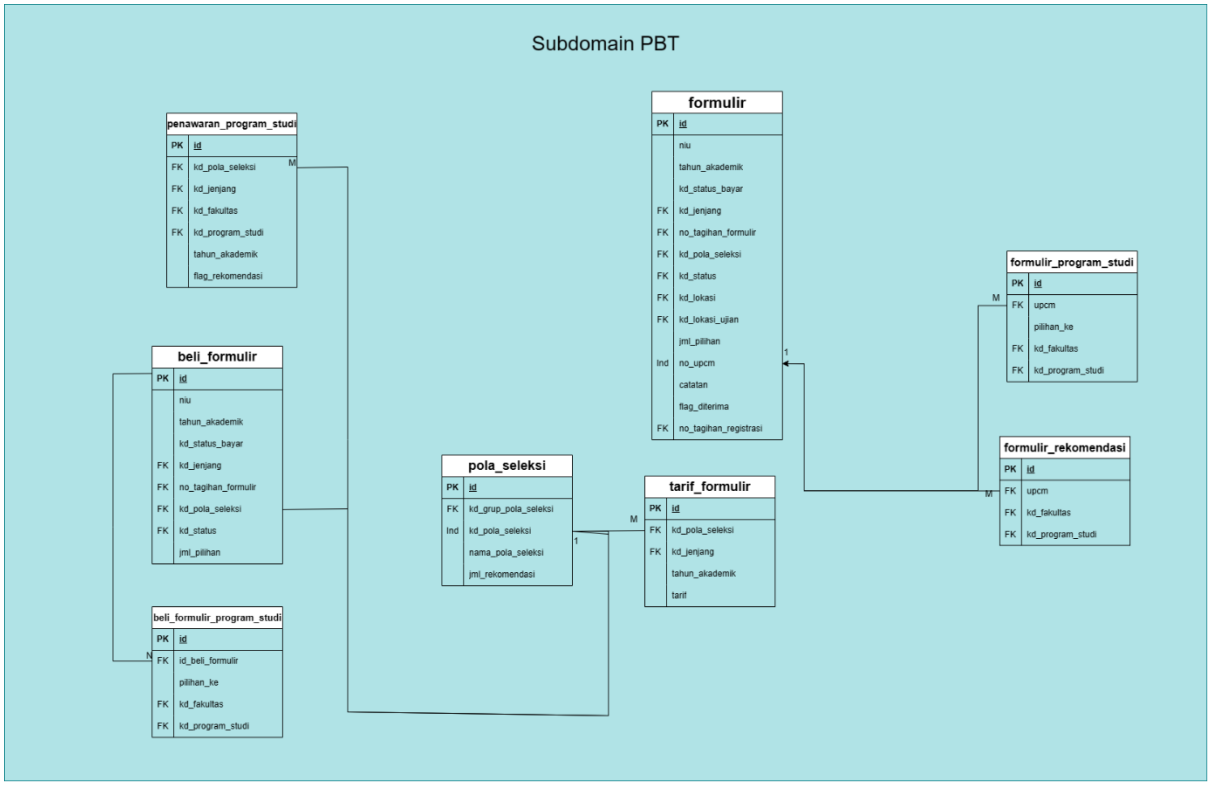
Gambar 3.30 ERD dari *microservice* PSB



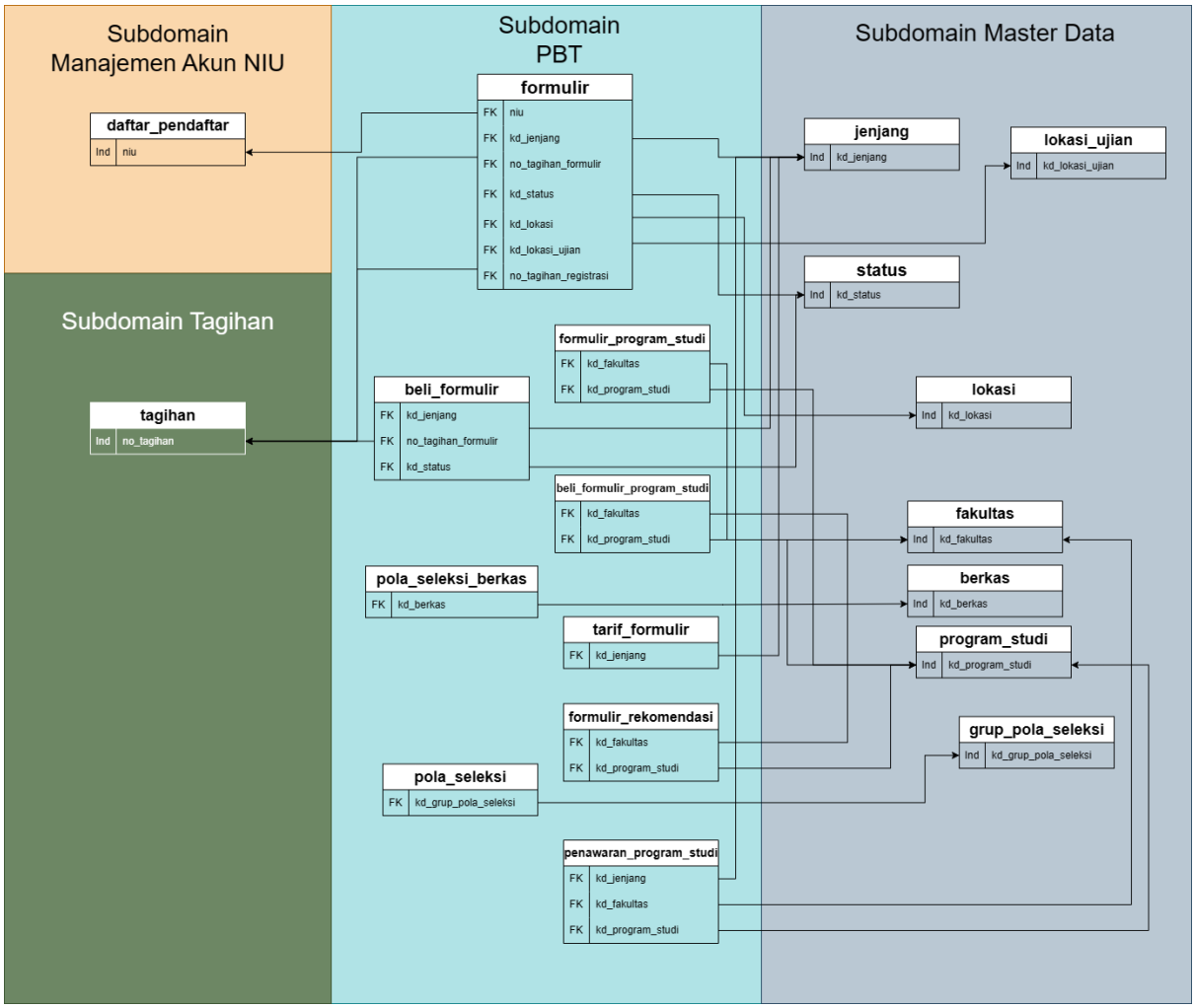
Gambar 3.31 Mapping hubungan tabel *subdomain* PSB

g. *Microservice* PBT

Microservice PBT memiliki tabel yang sama dengan *microservice* CBT dan struktur tabel yang sama dengan *microservice* CBT. Gambar 3.32 adalah ERD subdomain *PBT* dan Gambar 3.33 adalah pemetaan hubungan tabel subdomain *PBT* dengan subdomain lain.



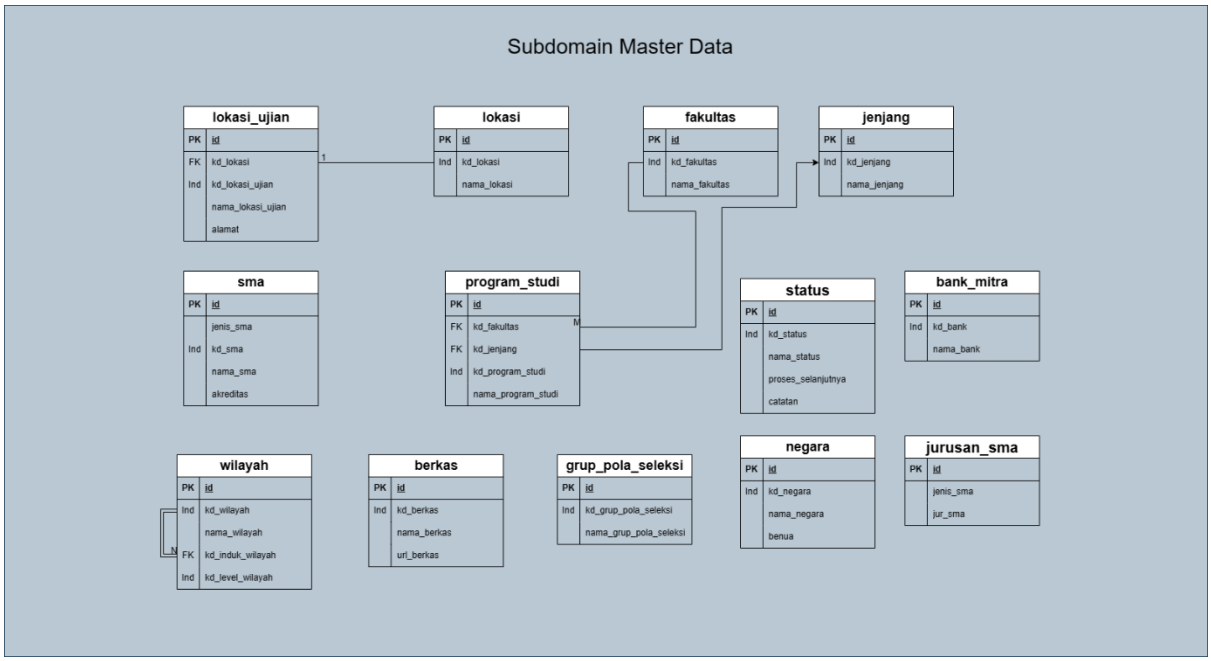
Gambar 3.32 ERD dari *microservice* PBT



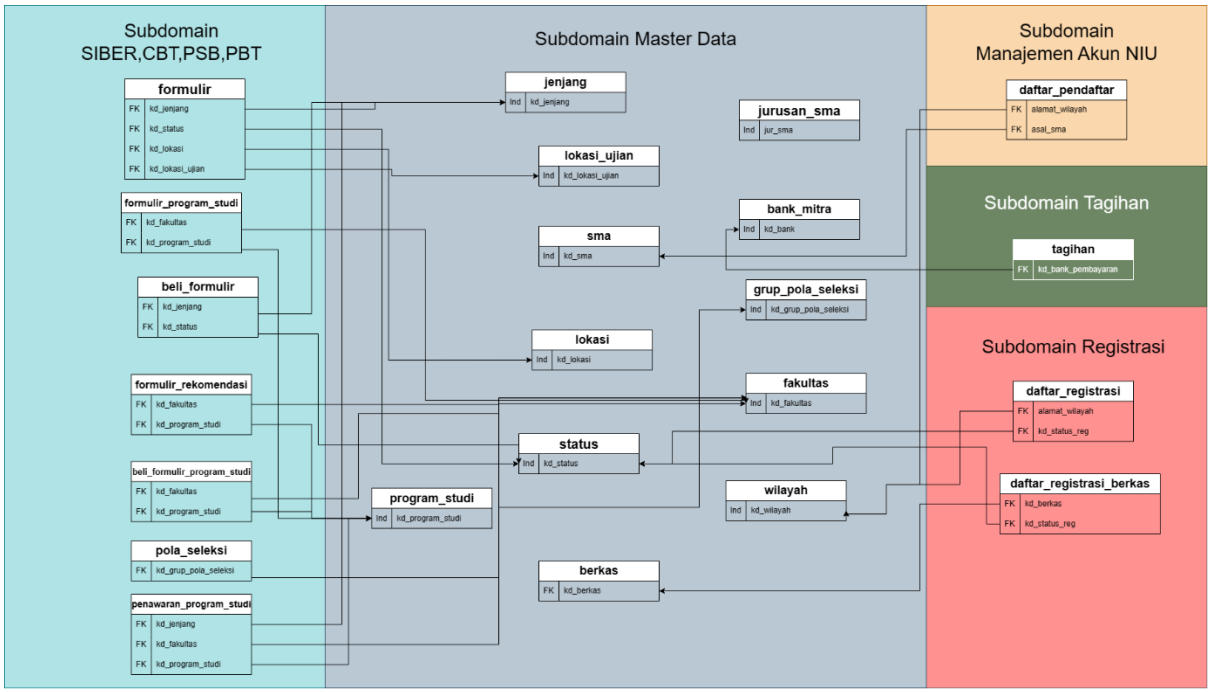
Gambar 3.33 Mapping hubungan tabel *subdomain* PBT

h. *Microservice* data *master*

Data yang berada pada *database* data *master* akan digunakan oleh banyak *microservice*. Pada *microservice* data *master* terdapat tabel bank_mitra, grup_pola_seleksi, fakultas, jenjang, jurusan_sma, lokasi, lokasi_ujian, status, sma, negara, wilayah dan program_studi. Struktur dari tabel dapat dilihat dari Tabel 3.20 hingga Tabel 3.31. Gambar 3.34 adalah ERD subdomain master data dan Gambar 3.35 adalah pemetaan hubungan tabel subdomain *master data* dengan subdomain lain.



Gambar 3.34 Mapping hubungan tabel *subdomain* Master Data



Gambar 3.35 Mapping hubungan tabel *subdomain* master data

Tabel 3.20 Struktur tabel bank_mitra

Kolom	Tipe data
id	int
kd_bank	varchar
nama_bank	varchar

Tabel 3.21 Struktur tabel grup_pola_seleksi

Kolom	Tipe data
id	int
kd_grup_pola_seleksi	varchar
nama_grup_pola_seleksi	varchar

Tabel 3.22 Struktur tabel fakultas

Kolom	Tipe data
id	int
kd_fakultas	varchar
nama_fakultas	varchar

Tabel 3.23 Struktur tabel jenjang

Kolom	Tipe data
id	int
kd_jenjang	varchar
nama_jenjang	varchar

Tabel 3.24 Struktur tabel jurusan_sma

Kolom	Tipe data
id	int
jenis_sma	varchar

jur_sma	varchar

Tabel 3.25 Struktur tabel lokasi

Kolom	Tipe data
id	int
kd_lokasi	varchar
nama_lokasi	varchar

Tabel 3.26 Struktur tabel lokasi_ujian

Kolom	Tipe data
id	int
kd_lokasi	varchar
kd_lokasi_ujian	varchar
nama_lokasi_ujian	varchar
alamat	varchar

Tabel 3.27 Struktur tabel status

Kolom	Tipe data
id	int
kd_status	varchar
nama_status	varchar
proses_selanjutnya	varchar
catatan	varchar

Tabel 3.28 Struktur tabel sma

Kolom	Tipe data
id	int
kd_sma	varchar

nama_sma	varchar
jenis_sma	varchar
akreditas	varchar

Tabel 3.29 Struktur tabel negara

Kolom	Tipe data
id	int
kd_negara	varchar
nama_negara	varchar
benua	varchar

Tabel 3.30 Struktur tabel wilayah

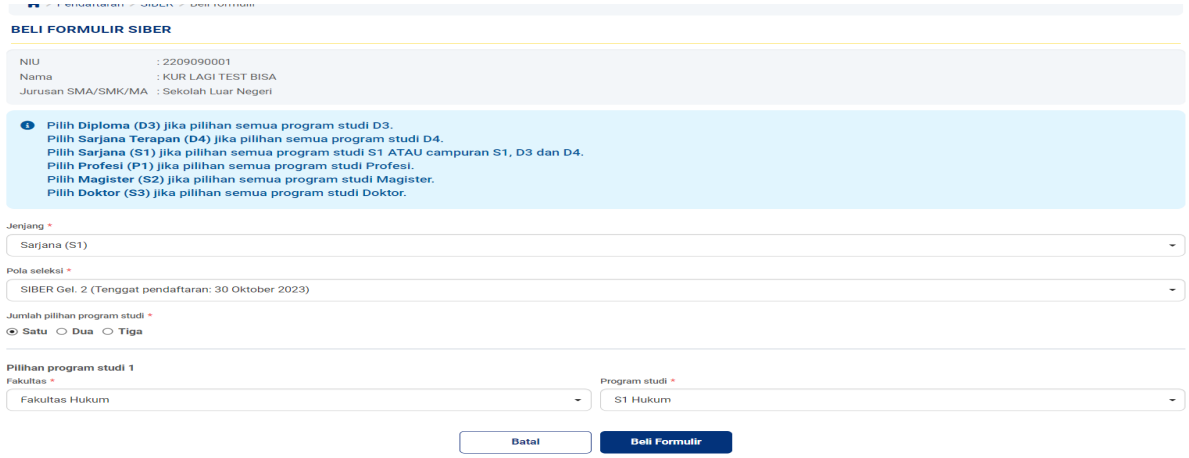
Kolom	Tipe data
id	int
kd_wilayah	varchar
nama_wilayah	varchar
kd_induk_wilayah	varchar
kd_level_wilayah	varchar

Tabel 3.31 Struktur tabel program studi

Kolom	Tipe data
id	int
kd_program_studi	varchar
nama_program_studi	varchar
kd_fakultas	varchar
kd_jenjang	varchar

Beli formulir

Fitur ini merupakan fitur membeli formulir pendaftaran di salah satu pola seleksi yang tanggal pendaftarannya sudah dibuka. Untuk dapat membeli formulir pendaftar harus memilih jenjang, pola seleksi dan program studi, seperti pada Gambar 3.20. Untuk memenuhi kebutuhan tersebut maka dibuatlah beberapa *endpoint* untuk dapat mengakses informasi yang dibutuhkan agar fitur beli formulir dapat digunakan. *Endpoint* yang dibutuhkan pada beli formulir dapat dilihat pada Tabel 3.32. Merujuk pada Tabel 3.32, hasil dari pembuatan *endpoint* 1 dan 2 dapat dilihat pada Gambar 3.21 dan Gambar 3.22. Untuk *endpoint* 3, *response* dari *endpoint* tersebut adalah nomor tagihan yang terbuat lalu data formulir pendaftaran akan tersimpan di *database* dan sebuah tagihan akan muncul pada halaman daftar tagihan.

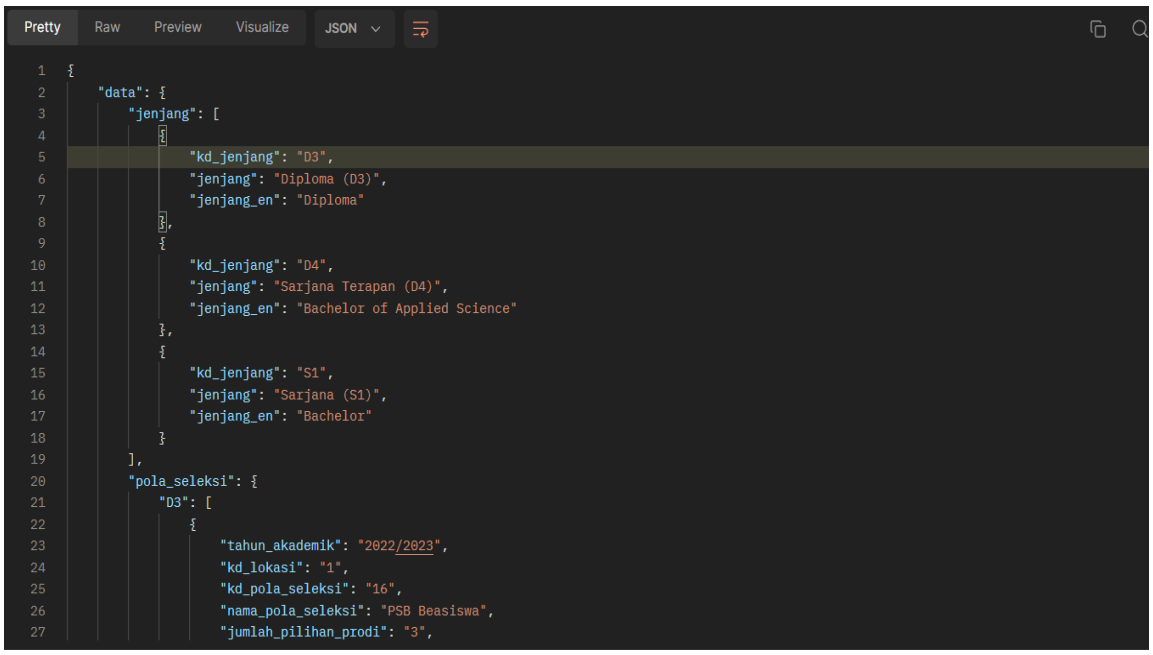


Gambar 3.36 Tampilan halaman beli formulir

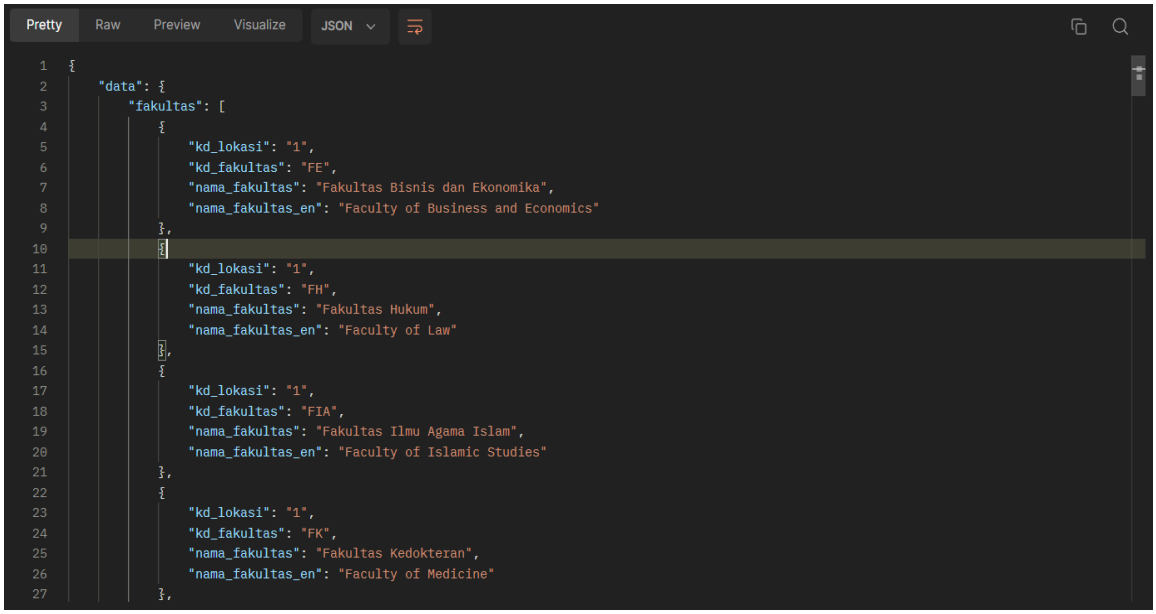
Tabel 3.32 Daftar *endpoint* untuk fitur beli formulir

No	Endpoint	Method	Deskripsi
1	(SIBER/CBT/PSB/PBT)/form/selection-pattern	GET	*Endpoint* untuk mendapatkan daftar informasi pola seleksi yang terbuka pada tanggal *request* dilakukan.
2	(SIBER/CBT/PSB/PBT)/form/organization	GET	*Endpoint* untuk mendapatkan daftar informasi program studi dan fakultas mana saja yang ditawarkan pada pola seleksi yang dipilih.

3	(SIBER/CBT/PSB/PBT)/purchase-form	POST	*Endpoint* untuk membuat formulir pendaftaran dengan program studi dan pola seleksi yang dipilih.



Gambar 3.37 *Response* dari *endpoint* GET pola seleksi



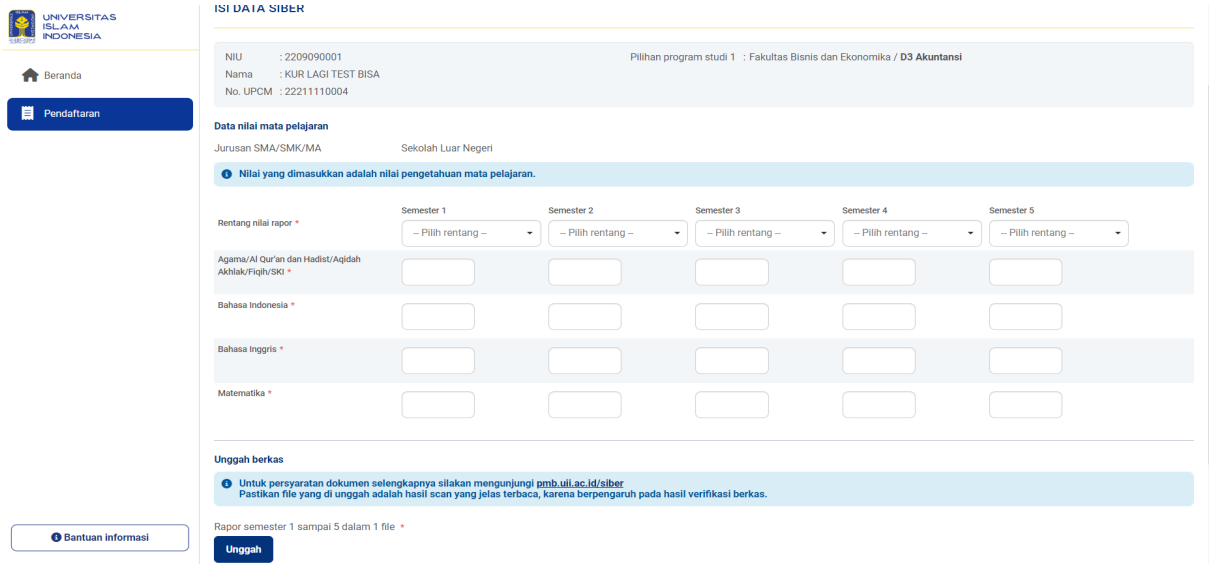
Gambar 3.38 *Response* dari *endpoint* GET program studi dan fakultas

Isi Data Formulir

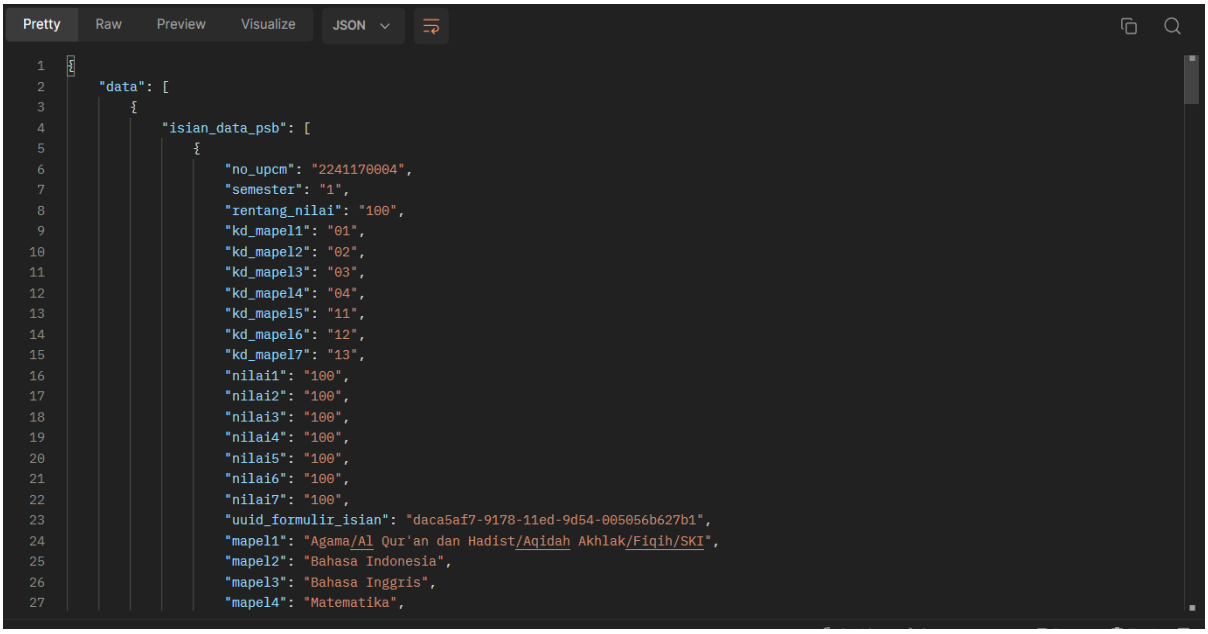
Fitur ini merupakan fitur untuk mengisi, melihat, mengubah, dan memperbaiki isian data pada suatu formulir pendaftaran milik pendaftar. Data isian merupakan data yang diperlukan pada masing-masing pola seleksi, seperti pada pola seleksi SIBER isian data berupa isi data rapor SMA/sederajat pendaftar dari semester satu sampai semester lima. Tampilan dari halaman isian data pendaftar terdapat pada Gambar 3.23. Untuk memenuhi kebutuhan tersebut maka dibuatkanlah beberapa *endpoint* untuk menyimpan isian data pendaftar, mengedit isian data pendaftar dan memberikan informasi isian data pendaftar. *Endpoint* untuk fitur ini dapat dilihat pada Tabel 3.32. Merujuk pada Tabel 3.32, *response* dari *endpoint* 1 dapat dilihat pada Gambar 3.24. *Response* dari *endpoint* 2 dan 3 adalah HTTP *status code* 200 ketika proses mengisi atau mengubah isian data berhasil disimpan. Server akan mengirimkan HTTP *status code* 400 ketika terjadi kesalahan klien, seperti data kurang ketika klien melakukan *request* dan server akan mengirimkan HTTP *status code* 500 ketika terjadi kesalahan pada server.

Tabel 3.33 Daftar *endpoint* untuk fitur isi data formulir

No	Endpoint	Method	Deskripsi
1	(SIBER/CBT/PSB/PBT)/fill-out-form/	GET	*Endpoint* untuk mendapatkan informasi terkait isian data pendaftar. *Endpoint* ini akan digunakan ketika pendaftar ingin mengubah data isian.
2	(SIBER/CBT/PSB/PBT)/fill-out-form/	PUT	*Endpoint* mengubah data isian pendaftar.
3	(SIBER/CBT/PSB/PBT)/fill-out-form/	POST	*Endpoint* untuk mengirim data isian pendaftar.



Gambar 3.39 Halaman tampilan isi data formulir



Gambar 3.40 *Response* dari *endpoint* GET data isian formulir

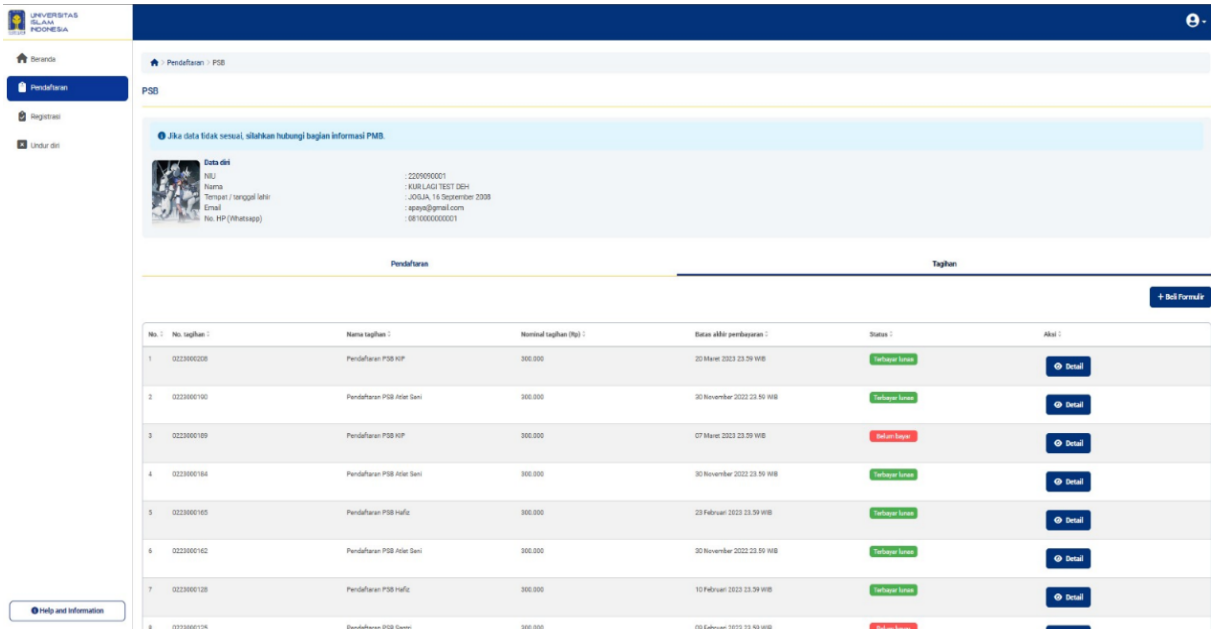
Tagihan

Fitur ini digunakan untuk melihat daftar tagihan milik pendaftar secara ringkas, seperti pada Gambar 3.25, isi dari data tersebut meliputi nomor tagihan, nominal yang harus dibayar, informasi terkait pola seleksi, bank mitra atau lokasi tempat pembayaran, status pembayaran, program studi pada formulir pendaftaran, tenggat pembayaran, tanggal pembayaran dilakukan dan NIU dari pemiliki tagihan. Fitur ini juga termasuk memberikan informasi tentang satu tagihan secara detil, untuk kebutuhan itu dibuatkanlah dua *endpoint* untuk memberikan daftar

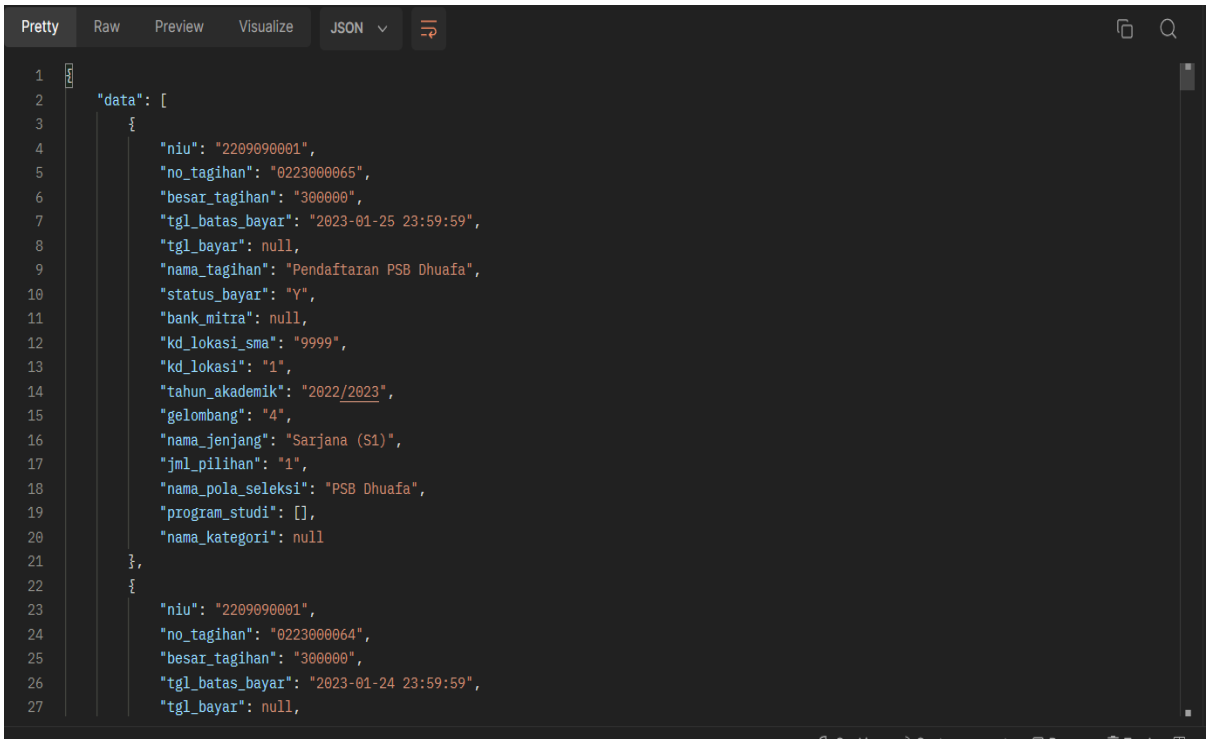
tagihan dan detail satu tagihan. *Endpoint* pada fitur ini dapat dilihat pada tabel 3.33. *Response* dari kedua *endpoint* tersebut dapat dilihat pada Gambar 3.26 dan 3.27.

Tabel 3.34 Daftar *endpoint* untuk fitur tagihan

No	Endpoint	Method	Deskripsi
1	(Tagihan)/bill/	GET	*Endpoint* untuk mendapatkan informasi daftar seluruh tagihan milik pendaftar.
2	(Tagihan)/bill/{id}	GET	*Endpoint* untuk mendapatkan informasi terkait hanya satu tagihan.



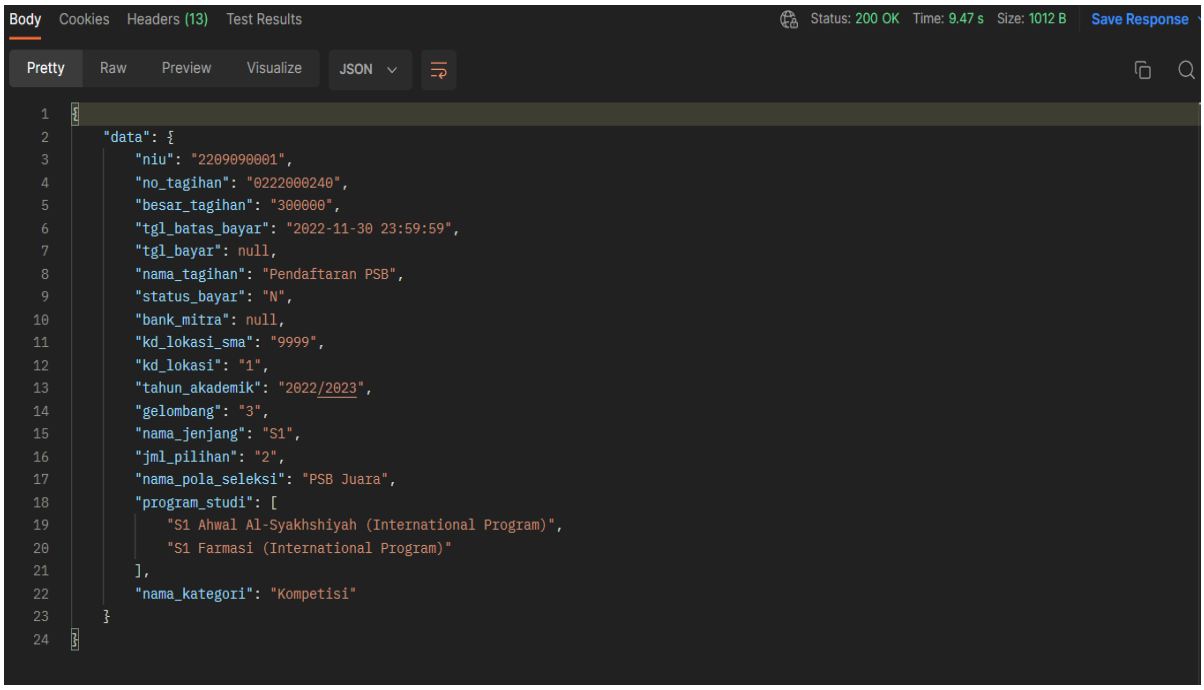
Gambar 3.41 Tampilan halaman daftar tagihan



```
Pretty Raw Preview Visualize JSON

{
    "data": [
        {
            "niu": "2209090001",
            "no_tagihan": "0223000065",
            "besar_tagihan": "300000",
            "tgl_batas_bayar": "2023-01-25 23:59:59",
            "tgl_bayar": null,
            "nama_tagihan": "Pendaftaran PSB Dhuafa",
            "status_bayar": "Y",
            "bank_mitra": null,
            "kd_lokasi_sma": "9999",
            "kd_lokasi": "1",
            "tahun_akademik": "2022/2023",
            "gelombang": "4",
            "nama_jenjang": "Sarjana (S1)",
            "jml_pilihan": "1",
            "nama_pola_seleksi": "PSB Dhuafa",
            "program_studi": [],
            "nama_kategori": null
        },
        {
            "niu": "2209090001",
            "no_tagihan": "0223000064",
            "besar_tagihan": "300000",
            "tgl_batas_bayar": "2023-01-24 23:59:59",
            "tgl_bayar": null,
```


Gambar 3.42 *Response* dari *endpoint* GET daftar tagihan



```
Body Cookies Headers (13) Test Results Status: 200 OK Time: 9.47 s Size: 1012 B Save Response

Pretty Raw Preview Visualize JSON

{
    "data": {
        "niu": "2209090001",
        "no_tagihan": "0222000240",
        "besar_tagihan": "300000",
        "tgl_batas_bayar": "2022-11-30 23:59:59",
        "tgl_bayar": null,
        "nama_tagihan": "Pendaftaran PSB",
        "status_bayar": "N",
        "bank_mitra": null,
        "kd_lokasi_sma": "9999",
        "kd_lokasi": "1",
        "tahun_akademik": "2022/2023",
        "gelombang": "3",
        "nama_jenjang": "S1",
        "jml_pilihan": "2",
        "nama_pola_seleksi": "PSB Juara",
        "program_studi": [
            "S1 Ahwal Al-Syakhshiyah (International Program)",
            "S1 Farmasi (International Program)"
        ],
        "nama_kategori": "Kompetisi"
    }
}
```


Gambar 3.43 *Response* dari *endpoint* GET detil tagihan

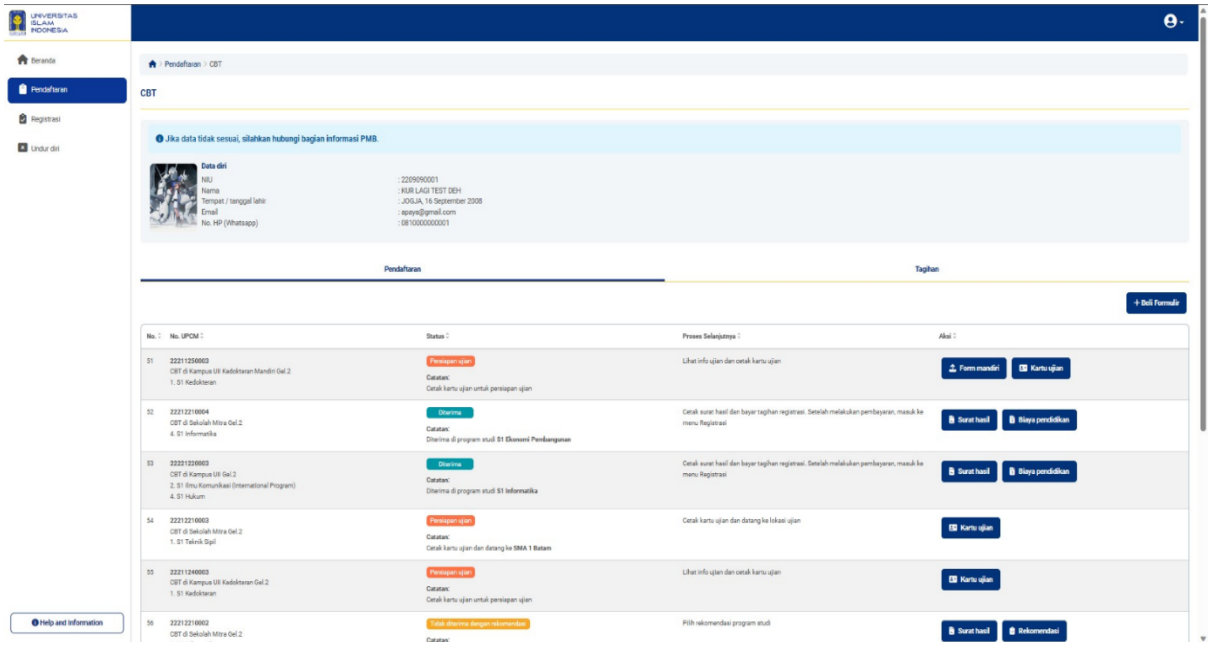
List Formulir Pendaftaran

Fitur ini berfungsi untuk menampilkan data daftar formulir pendaftar yang tagihannya formulirnya telah terbayar. Pada data tersebut akan berisi informasi terkait status, proses

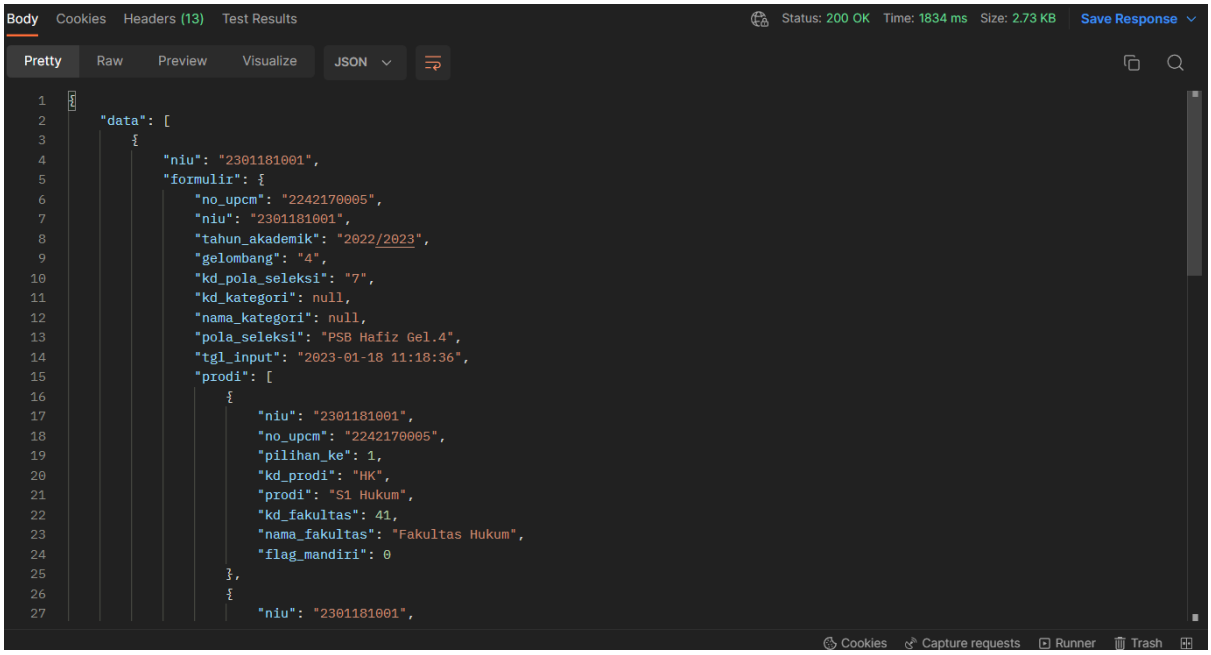
selanjutnya yang pendaftar harus lakukan, informasi terkait pola seleksi yang didaftar, tanggal input agar *list* tersebut dapat diurutkan, dan juga dokumen-dokumen yang pendaftar akan dapatkan pada status tertentu, seperti dokumen surat hasil diterima dan dokumen kartu ujian di tempat mitra. Halaman dari fitur ini dapat dilihat pada Gambar 3.28. Untuk kebutuhan itu maka dibuatkan hanya satu *endpoint* untuk memberikan data daftar formulir pendaftaran milik pendaftar. *Endpoint* dapat dilihat pada Tabel 3.34 dan *Response* dari *endpoint* tersebut dapat dilihat pada Gambar 3.29.

Tabel 3.35 Endpoint fitur *list* pendaftaran

No	Endpoint	Method	Deskripsi
1	(SIBER/CBT/PSB/PBT)/list-admission	GET	*Endpoint* untuk mendapatkan informasi daftar formulir pendaftaran milik pendaftar yang tagihan formulirnya telah dibayar.



Gambar 3.44 Halaman daftar formulir pendaftaran



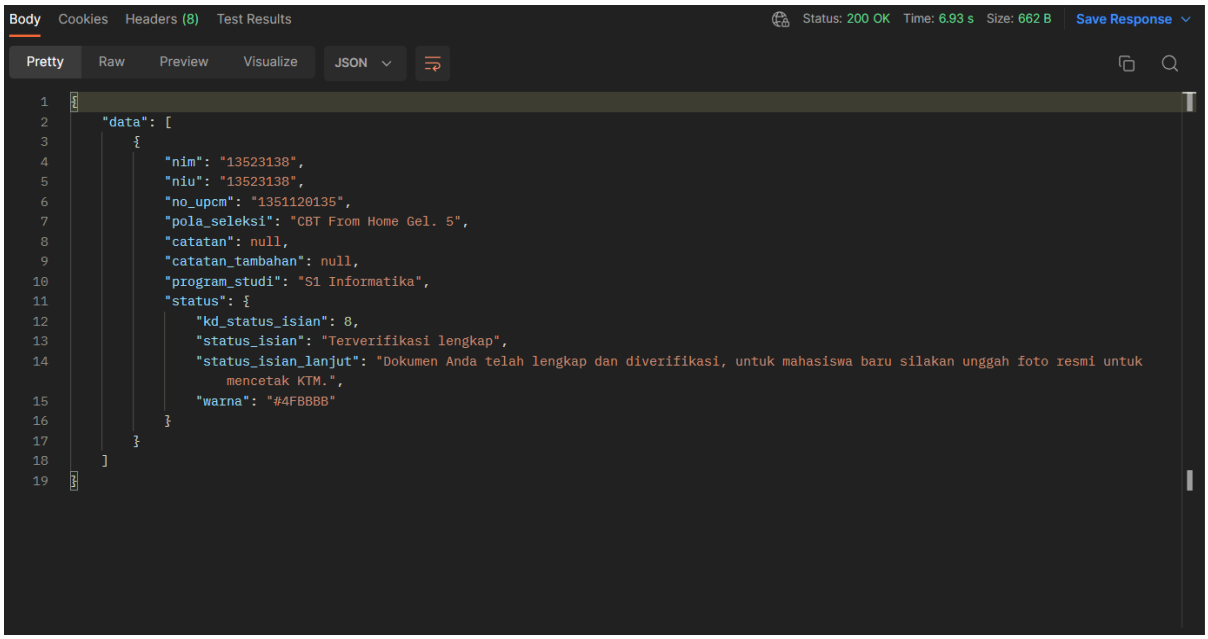
Gambar 3.45 *Response* dari *endpoint* GET data daftar formulir pendaftaran

List Registrasi

Fitur ini merupakan fitur untuk menampilkan semua data registrasi. Pendaftar akan mulai menggunakan fitur ini ketika pendaftar memiliki formulir pendaftaran yang statusnya telah diterima di salah satu program studi dan sudah membayar tagihan registrasi. Registrasi yang dimaksud adalah proses pendaftar mulai memasukkan data diri untuk disimpan ke sistem akademik sebagai mahasiswa, pendaftar kemudian akan mendapatkan Nomor Induk Mahasiswa yang akan digunakan selama kegiatan perkuliahan di UII. Untuk kebutuhan tersebut maka dibuatkanlah satu *endpoint* untuk memberikan data daftar registrasi, status registrasi tersebut dan proses yang harus dilakukan oleh pendaftar. *Endpoint* yang dibuat dapat dilihat pada Tabel 3.35 dan *response* dari *endpoint* ini dapat dilihat pada Gambar 3.30.

Tabel 3.36 *Endpoint* fitur *list* registrasi

No	Endpoint	Method	Deskripsi
1	(Registrasi)/registration-list	GET	*Endpoint* untuk mendapatkan informasi daftar registrasi milik pendaftar beserta status dan proses selanjutnya.



Gambar 3.46 *Response* dari *endpoint* GET *list* registrasi

Rekomendasi

Fitur ini merupakan fitur untuk memilih program studi yang direkomendasikan oleh sistem ketika pendaftar tidak diterima di semua program studi yang dipilih ketika pada formulir pendaftaran, fitur ini memberikan kesempatan kepada pendaftar untuk memilih program studi lain yang ditawarkan oleh sistem, jika pendaftar memilih salah satu program studi yang ditawarkan, pendaftar akan langsung diterima pada program studi tersebut. Sistem akan membuat *list* program studi yang direkomendasikan berdasarkan dengan nilai *passing grade* pendaftar. *Passing grade* merupakan total poin yang didapatkan dari rumus yang sudah ditentukan, masing-masing program studi memiliki batas minimal *passing grade*, seorang pendaftar harus memiliki nilai *passing grade* di atas nilai minimal *passing grade* pada program studi tersebut untuk dapat diterima pada program studi tersebut. Untuk kebutuhan itu maka dibuatkan dua *endpoint* untuk mendapatkan daftar rekomendasi program studi dan *endpoint* untuk mengirimkan pilihan program studi yang pendaftar pilih. *Endpoint* untuk fitur rekomendasi dapat dilihat pada Tabel 3.36.

Tabel 3.37 Daftar *endpoint* untuk fitur rekomendasi

No	Endpoint	Method	Deskripsi
1	(SIBER/CBT/PSB/PBT)/recommendation-form	GET	*Endpoint* untuk mendapatkan data daftar

			rekomendasi program studi pendaftar.
2	(SIBER/CBT/PSB/PBT)/recommendation-form	POST	*Endpoint* mengirimkan data pilihan rekomendasi program studi yang dipilih oleh pendaftar.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Setelah selesai melakukan kegiatan magang selama delapan bulan, terdapat beberapa perbedaan antara teori dengan yang terjadi di lapangan. Secara teori, penerapan *clean architecture* dapat membuat sistem yang dibangun menjadi *Independent of Framework*, *Testable*, *Independent of UI*, dan *Independent of Database*. Namun, pada saat praktek terdapat pengecualian seperti yang terjadi pada saat pengembangan *back-end* UIIAdmisi. *Back-end* UIIAdmisi dibuat dengan menggunakan *framework* Laravel Lumen, *framework* tersebut membungkus sistem secara keseluruhan dan juga beberapa fungsional pada sistem bergantung pada fitur-fitur yang ada pada Laravel Lumen, seperti pada *data access layer* yang bergantungan dengan fitur Eloquent yang memudahkan interaksi dengan basis data sehingga sistem *back-end* UIIAdmisi tidak sepenuhnya independen dari *framework*. *Independent of Framework* pada *clean architecture* dapat dicapai ketika *framework* hanya membungkus pada suatu fungsional saja, seperti *framework* Echo pada bahasa pemrograman Golang yang memiliki fungsi sebagai *framework* web, di saat *framework* Echo diganti dengan *framework* web yang lain, fungsionalitas lain seperti pada *data access layer* tidak akan terganggu.

4.2 Pembelajaran Magang

Setelah menjalani magang di Badan Sistem Informasi UII selama delapan bulan, banyak sekali pembelajaran yang didapat, antara lain:

a. Selama melaksanakan magang mendapatkan pengalaman mengikuti *scrum* dan mempelajari bahwa scrum dapat adaptif di saat terjadi perubahan, seperti pada saat pengerjaan UIIAdmisi pernah terjadi perubahan pada aturan bisnis PMB dua minggu sebelum UIIAdmisi mulai digunakan oleh *end-user* yang mengharuskan banyak perubahan pada sistem seperti arsitektur data dan logika pada *microservice*. Namun, karena menerapkan *scrum* tim Admisi dapat beradaptasi dan membuat rencana pengembangan di saat *sprint planning* sehingga perubahan aturan bisnis tersebut dapat diterapkan tepat waktu.
b. Dari pelaksanaan magang mempelajari teori prinsip *clean architecture* dan mengapa prinsip tersebut banyak digunakan pada aplikasi-aplikasi di tempat magang. Selama praktek juga dapat dirasakan keunggulan dari implementasi *clean architecture*, seperti

pemisahan fungsional yang menjadi beberapa *layer* memudahkan proses *error tracking* di saat testing menjadi lebih mudah.

c. Selama magang juga mendapatkan pengetahuan terkait cara *version control* secara semantik dengan menggunakan Gitlab sebagai repositori tempat kode disimpan.

4.3 Hambatan dan Kendala Magang

Salah satu hambatan yang dirasakan selama magang adalah kurangnya akses ke *tools* manajemen proyek dan beberapa proses pengembangan. Dalam teori *scrum*, terdapat beberapa nilai yang menjadi junjungan salah satunya adalah transparansi, transparansi dapat berarti transparansi pada isu apa saja yang akan dikerjakan dan siapa saja yang mengerjakan. Namun, dengan keterbatasan akses ke beberapa *tools* manajemen proyek mengakibatkan berkurangnya nilai tersebut.

Tantangan yang dirasakan saat aktivitas magang adalah mempelajari teori prinsip *clean architecture* dan mengapa prinsip tersebut digunakan pada aplikasi-aplikasi yang akan dikembangkan. Teori tersebut tidak diajarkan pada saat kuliah, tetapi konsep tersebut telah banyak digunakan dalam industri. Mempelajari teori tersebut dan langsung mengimplementasikan teori tersebut pada saat proses pengembangan merupakan tantangan yang cukup sulit untuk dilakukan.

Tantangan lain yang dirasakan adalah harus bisa melakukan manajemen waktu dengan baik, dikarenakan pelaksanaan magang selama lima bulan pertama juga bersamaan dengan melaksanakan kelas perkuliahan yang mana jam pelaksanaan kelas tersebut bersamaan dengan jam kerja di kantor. Tantangan tersebut dapat diselesaikan dengan cara mengkomunikasikan hal tersebut dengan pihak *scrum master* di Badan Sistem Informasi UII untuk meminta waktu izin agar dapat mengikuti kelas perkuliahan tersebut dengan syarat harus mengganti waktu kerja di lain waktu.

BAB V
PENUTUP

5.1 Kesimpulan

Pengembangan ulang sisi *back-end* sistem UIIAdmisi pada fitur beli formulir, *list* pendaftaran, tagihan, rekomendasi, *list* registrasi, dan isi data formulir berhasil dilakukan dengan memanfaatkan *microservices* yang menggunakan pendekatan *domain driven design* untuk desain microservice dan juga menerapkan konsep *clean architecture* dalam menstrukturkan kode pada *microservice*. Dengan memanfaatkan pendekatan *domain driven design*, proses pengembangan fitur berikutnya akan menjadi lebih mudah, seperti ketika ada pola seleksi baru yang akan digunakan di UIIAdmisi, tim pengembang dapat membuat *microservice* dan basis data baru sehingga proses pengembangan *microservice* baru ini tidak akan mengganggu *microservice* untuk pola seleksi yang lain.

Untuk memenuhi kebutuhan pada fitur beli formulir maka dibuat tiga *endpoint* untuk menyediakan informasi pola seleksi yang sedang dibuka dan program studi yang ditawarkan serta *endpoint* untuk membuat formulir pendaftaran tersebut. Untuk fitur *list* pendaftaran dihasilkan satu *endpoint* untuk memberi informasi daftar formulir pendaftaran milik pendaftar untuk ditampilkan kedalam bentuk tabel pada sisi *front-end*. Pada fitur tagihan dibuat dua *endpoint* untuk memberi daftar tagihan milik pendaftar dan memberi satu informasi tagihan secara rinci. Pada fitur rekomendasi dibuat dua *endpoint* untuk memberi informasi daftar program studi yang dianjurkan oleh sistem dan mengirim informasi program studi rekomendasi yang telah dipilih oleh pendaftar ke sistem. Untuk fitur *list* registrasi dibuat hanya satu *endpoint* untuk memberikan daftar formulir pendaftaran yang telah memasuki tahap registrasi ke induk sistem BSI UII ketika pendaftar telah diterima pada program studi dan membayar tagihan registrasi. Untuk fitur isi data formulir dibuatkan tiga *endpoint* yang digunakan untuk mengirim isian data formulir ketika pendaftar pertama kali melakukan isi data formulir, meminta informasi isian data formulir yang telah diisi oleh pendaftar, dan mengubah isi data formulir ketika pendaftar melakukan perbaikan.

5.2 Saran

Terdapat beberapa saran yang dapat diterapkan pada pengembangan *back-end* UIIAdmisi berikutnya, salah satunya adalah mengurangi ketergantungan terhadap *framework*, seperti yang telah dibahas pada bab sebelumnya, salah satu kelemahan yang ada pada *back-end* UIIAdmisi

adalah ketergantungan pada *framework* Laravel Lumen seperti yang sudah dibahas pada bab sebelumnya. Saran lainnya untuk pengembangan *back-end* UIIAdmisi berikutnya adalah mengurangi keterikatan antara *class* pada masing-masing *layer* dengan mengimplementasikan *abstract class* yang berfungsi untuk menghubungkan antara *layer*.

DAFTAR PUSTAKA

Bauroziq. (2022, July 14). *Belajar SCRUM, Pengertian, Event, Artefact, Role, Team, dan kelebihannya.* Diambil kembali dari caraguna: https://caraguna.com/belajar-scrum/

Fajri, A. R., & Rani, S. (2022). Penerapan Design Pattern MVVM dan Clean Architecture pada Pengembangan Aplikasi Android (Studi Kasus: Aplikasi Agree Partner). *AUTOMATA*.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Irvine: University of California.

Hadji, S., Taufik, M., & Mulyono, S. (2019). IMPLEMENTASI METODE SCRUM PADA PENGEMBANGAN APLIKASI DELIVERY ORDER BERBASIS WEBSITE (STUDI KASUS PADA RUMAH MAKAN LOMBOK IDJO SEMARANG). *PROSIDING KONFERENSI ILMIAH MAHASISWA UNSSULA (KIMU) 2.* Semarang.

Harris, C. (t.thn.). *Agile scrum artifacts*. Diambil kembali dari Atlassian: https://www.atlassian.com/agile/scrum/artifacts

Leitch, R. A., & Davis, K. R. (1992). *Accounting Information Systems: Theory and Practice.* Prentice Hall.

Mozilla Developer. (t.thn.). *HTTP Messages*. Diambil kembali dari MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages#http_requests

Nadareushvuku, I., Mitra, R., Mclarty, M., & Amundsen, M. (2016). *Microservice Architecture: Aligning Principles, Practice, and Culture.* Sebastopol: O'Reilly Media, Inc.

Nuralamsyah, Akbar, R. J., & Fabroyir, H. (2021). Rancang Bangun Modul Job Marketplace di Aplikasi MyITS Connect Berdasarkan Onion Architecture dengan Paradigma Domain Driven Design. *Jurnal Teknik ITS, 10*(2), A99-A105.

Paramitha, I. A., Wiharta, D. M., & Suyadna, I. A. (2022). PERANCANGAN DAN IMPLEMENTASI RESTFUL API PADA SISTEM INFORMASI MANAJEMEN DOSEN UNIVERSITAS UDAYANA. *Jurnal SPEKTRUM, 9*(3), 15-23.

Pawana, I. A., Wiharta, D. M., & Sastra, N. P. (2021). Identifikasi Kandidat Microservies Dengan Analisis Domain Driven Design. *Majalah Ilmiah Teknologi Elektro, 20*(2), 273-279.

Perdana, M. A. (2018). Pengembangan REST API Layanan Penyimpanan Menggunakan Metode Rapid Application Development (Studi Kasus: PT. XYZ). *Jurnal Nasional Informatika dan Teknologi Jaringan*, 100-104.

Ramadhini, Y. A., & Paputungan, I. V. (2022). Improvisasi Task pada Software Manajemen Proyek (Studi Kasus: E-Commerce). *AUTOMATA*.

Sinambela, A., Ernawati, & Coastera, F. F. (2021). IMPLEMENTASI ARSITEKTUR MICROSERVICES PADA RANCANG BANGUN APLIKASI MARKETPLACE BERBASIS WEB. *Jurnal Rekursif, 9*(1), 1-13.

Suryotrisongko, H. (2017). Arsitektur Microservice untuk Resiliensi Sistem Informasi. *Jurnal Sisfo , 06*(02), 235-250.

LAMPIRAN