BAB III

METODOLOGI PENELITIAN DAN PENGEMBANGAN

3.1 Model Pengembangan Sistem: Model Prototipe

Untuk pengembangan aplikasi FETNET, model pengembangan sistem yang dipilih adalah Model Prototipe (*Prototyping Model*). Pemilihan model ini didasarkan pada sifat proyek yang sangat berorientasi pada pengguna akhir (staf administrasi akademik) dan kebutuhan untuk memvalidasi desain antarmuka yang kompleks secara iteratif. Model prototipe memungkinkan pengembang untuk membuat versi awal dari sistem dengan fungsionalitas terbatas namun dapat dioperasikan, yang kemudian disajikan kepada pengguna untuk mendapatkan umpan balik.

Justifikasi utama penggunaan model ini adalah sebagai berikut:

- Klarifikasi Kebutuhan Pengguna: Kebutuhan pengguna, terutama terkait cara paling intuitif untuk mendefinisikan batasan jadwal yang rumit, tidak selalu dapat didefinisikan secara lengkap di awal proyek. Prototipe memungkinkan pengguna untuk "merasakan" alur kerja dan memberikan umpan balik yang konkret untuk penyempurnaan.
- Mengurangi Risiko Kegagalan Desain: Dengan mendapatkan umpan balik di tahap awal, risiko membangun sistem yang secara fungsional benar tetapi sulit digunakan dapat diminimalkan.
- Fleksibilitas dalam Pengembangan: Model ini memungkinkan adanya penyesuaian selama siklus pengembangan, seiring dengan munculnya pemahaman yang lebih dalam tentang masalah dari interaksi dengan pengguna.

Siklus pengembangan dalam proyek ini melibatkan pembuatan prototipe awal, peninjauan oleh pengguna, pengumpulan umpan balik, dan penyempurnaan prototipe secara berulang hingga produk akhir memenuhi semua kebutuhan yang telah divalidasi.

3.2 Prosedur Pengembangan Sistem

Prosedur pengembangan sistem FETNET mengikuti langkah-langkah terstruktur yang diadaptasi dari model prototipe:

1. Analisis Kebutuhan dan Pengumpulan Data: Tahap awal melibatkan komunikasi intensif dengan pihak-pihak terkait di FPTI UPI, termasuk wawancara untuk memahami alur kerja manual dan analisis dokumen kurikulum.
2. Perancangan Sistem (Desain): Berdasarkan hasil analisis, dilakukan perancangan sistem yang mencakup perancangan arsitektur, basis data, dan antarmuka pengguna (UI/UX).
3. Implementasi Prototipe: Proses penulisan kode untuk merealisasikan desain menggunakan tumpukan teknologi TALL (Laravel dan Livewire).
4. Pengujian dan Evaluasi Prototipe: Setiap versi prototipe yang fungsional diuji secara internal dan dievaluasi bersama calon pengguna untuk mendapatkan umpan balik kualitatif.
5. Penyempurnaan dan Iterasi: Umpan balik dari pengguna digunakan untuk memperbaiki dan menyempurnakan prototipe. Siklus ini diulang beberapa kali.
6. Implementasi Final dan Studi Kasus: Setelah aplikasi dianggap matang, dilakukan implementasi skala penuh menggunakan data riil dari FPTI UPI untuk satu semester sebagai evaluasi akhir.

3.3 Arsitektur Sistem FETNET

Perancangan arsitektur aplikasi FETNET didasarkan pada tantangan komputasi fundamental yang melekat pada *University Course Timetabling Problem* (UCTP). Sebagai masalah optimisasi yang tergolong NP-hard, proses pencarian solusi penjadwalan dapat memakan waktu yang lama dan tidak dapat diprediksi, berkisar dari beberapa menit hingga berjam-jam, tergantung pada kompleksitas dan ketatnya batasan yang diberikan.[1] Aplikasi web konvensional yang beroperasi pada

siklus permintaan-respons (request-response) sinkron memiliki batas waktu (timeout) yang umumnya singkat, biasanya antara 30 hingga 120 detik. Menjalankan proses penjadwalan yang intensif secara sinkron akan menyebabkan kegagalan permintaan (request timeout) yang tidak dapat dihindari, menghasilkan pengalaman pengguna yang buruk dan sistem yang tidak andal.

Untuk mengatasi risiko fundamental ini, arsitektur sistem FETNET dirancang dengan pendekatan asinkron yang mengimplementasikan pola desain *Facade/Wrapper*, seperti yang diilustrasikan pada Gambar 3.1. Arsitektur ini secara sengaja memisahkan (decouples) antarmuka pengguna dari proses komputasi penjadwalan yang berat, memastikan aplikasi tetap responsif, andal, dan skalabel.

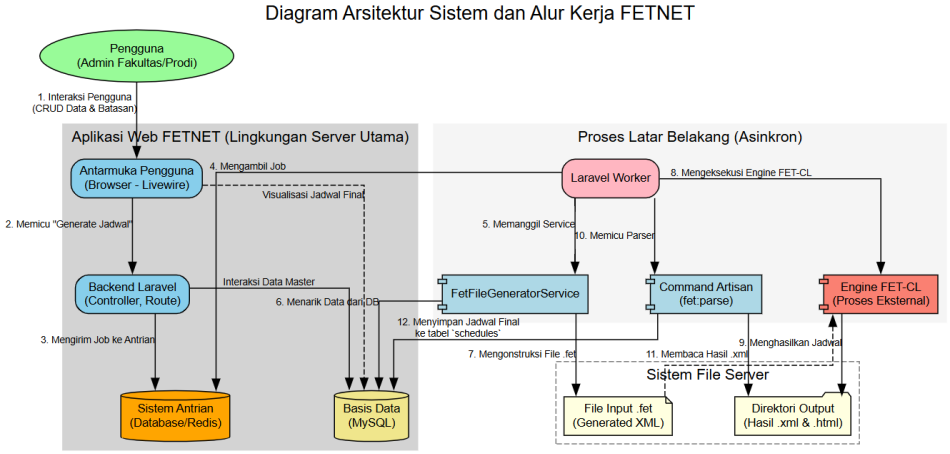
Alur kerja utama dalam arsitektur ini terbagi menjadi beberapa komponen logis yang saling berinteraksi. Pertama**,** Aplikasi Web (Facade/Wrapper) berfungsi sebagai antarmuka modern yang berinteraksi langsung dengan pengguna. Komponen ini, yang dibangun dengan tumpukan teknologi TALL, menyembunyikan seluruh kompleksitas interaksi dengan *engine* penjadwalan FET-CL, sehingga pengguna non-teknis dapat mendefinisikan masalah penjadwalan melalui formulir dan antarmuka visual yang intuitif.[1] Kedua, ketika pengguna memicu proses penjadwalan, permintaan tidak dieksekusi secara langsung. Sebaliknya, sebuah tugas (

job) dibuat dan dikirim ke dalam Sistem Antrian (Queue). Mekanisme ini adalah komponen vital yang secara efektif memisahkan permintaan pengguna yang singkat dari eksekusi tugas yang berpotensi berjalan lama. Ketiga, sebuah Proses Worker (Asinkron) yang berjalan secara independen di latar belakang akan mengambil tugas dari antrian dan mengeksekusinya. Proses ini bertanggung jawab untuk mengorkestrasi seluruh logika penjadwalan, mulai dari serialisasi data hingga eksekusi *engine* eksternal. Keempat, *worker* akan memanggil Engine FET-CL, yang diperlakukan sebagai "kotak hitam" komputasi. *Engine* ini menerima file input berformat .fet (XML) yang telah disiapkan, menjalankan algoritma heuristiknya, dan menghasilkan file output XML yang berisi solusi jadwal. Setelah *engine* selesai, *worker* akan mem-parsing file hasil dan menyimpan jadwal final ke dalam

basis data, yang kemudian dapat divisualisasikan oleh pengguna melalui antarmuka web.

Desain asinkron ini bukan sekadar pilihan teknis, melainkan sebuah strategi mitigasi risiko yang fundamental. Dengan memberikan respons instan kepada pengguna ("Proses penjadwalan telah dimulai") sambil membiarkan tugas berat berjalan aman di latar belakang tanpa terikat oleh batasan *timeout* HTTP, arsitektur ini secara inheren meningkatkan keandalan, toleransi kesalahan (*fault tolerance*), dan skalabilitas sistem secara keseluruhan.

Berikut pada Gambar 3.1 Arsitektur Sistem FETNET digambarkan bagaiman diagram yang mengilustrasikan arsitektur sistem FETNET. Diagram ini menyajikan gambaran tingkat tinggi dari arsitektur sistem FETNET, mengilustrasikan alur kerja asinkron dari interaksi pengguna di antarmuka web, pengiriman tugas ke sistem antrian, eksekusi oleh proses worker di latar belakang yang berinteraksi dengan engine FET-CL, hingga penyimpanan dan visualisasi hasil akhir.



Gambar 3 1 Arsitektur Sistem FETNET

3.4 Perancangan Basis Data

Pemilihan model basis data relasional, yang diimplementasikan

menggunakan MySQL, didasarkan pada justifikasi bahwa UCTP secara fundamental adalah masalah pengelolaan entitas-entitas diskrit (dosen, mata kuliah, ruangan) dengan hubungan yang terstruktur dan terdefinisi dengan baik.[1] Model relasional secara inheren unggul dalam merepresentasikan struktur semacam ini, memastikan integritas data, konsistensi, dan efisiensi kueri. Struktur basis data FETNET lebih dari sekadar repositori data; ia berfungsi sebagai formalisasi dari masalah optimisasi itu sendiri. Setiap tabel, kolom, dan relasi secara kolektif mendefinisikan ruang lingkup, variabel, dan batasan dari masalah UCTP spesifik FPTI yang akan diterjemahkan dan dipecahkan oleh *engine* FET.

Diagram Entitas-Hubungan (ERD) yang disajikan pada Gambar 3.2 merepresentasikan model data logis dari sistem FETNET. Penjelasan mendalam mengenai struktur basis data ini dapat diuraikan berdasarkan klaster entitas fungsionalnya. Entitas Dasar (Master Data) membentuk fondasi sistem, menyimpan data inti yang relatif statis. Ini mencakup users (pengguna sistem), roles (peran pengguna), prodis (program studi), teachers (dosen), subjects (mata kuliah), student_groups (kelompok mahasiswa), dan rooms (ruangan). Tabel-tabel ini berfungsi sebagai "kamus" data yang akan dirujuk oleh seluruh sistem.

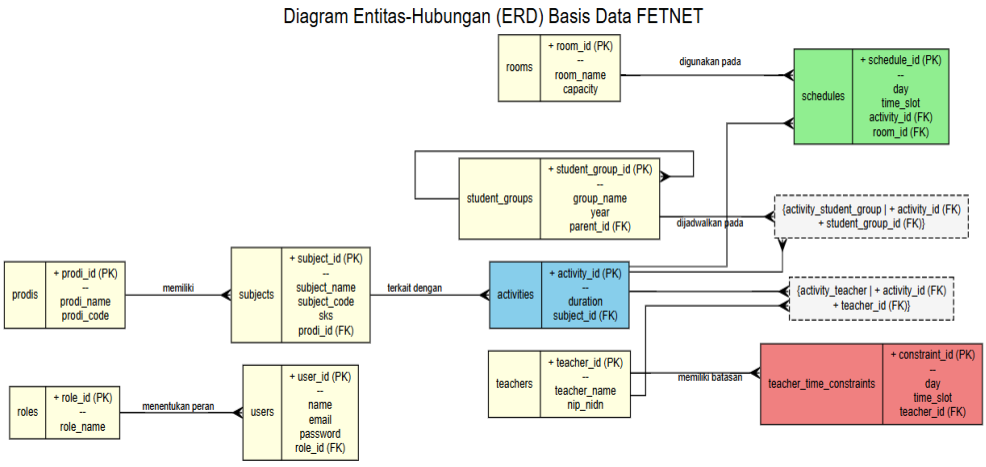
Jantung dari perancangan ini adalah Entitas Transaksional Inti, yaitu tabel activities. Tabel ini secara efektif merepresentasikan setiap unit perkuliahan unik yang harus dijadwalkan (misalnya, "Mata Kuliah Kalkulus oleh Dosen A untuk Kelas B"). Tabel ini menghubungkan semua entitas master melalui relasi, termasuk hubungan *many-to-many* yang diimplementasikan melalui tabel perantara seperti activity_teacher (satu aktivitas bisa diajar oleh tim dosen) dan activity_student_group (satu aktivitas bisa diikuti oleh kelas gabungan).

Selanjutnya, Entitas Batasan seperti teacher_time_constraints dan room_time_constraints berfungsi untuk merepresentasikan aturan-aturan bisnis dan preferensi secara digital. Data dalam tabel-tabel ini secara eksplisit mendefinisikan batasan keras (*hard constraints*) yang tidak boleh dilanggar oleh algoritma, seperti slot waktu di mana seorang dosen tidak bersedia mengajar.

Terakhir, Entitas Hasil, yaitu tabel schedules, berfungsi sebagai tujuan akhir

dari seluruh proses. Tabel ini pada awalnya kosong dan akan diisi dengan solusi jadwal yang valid setelah *engine* FET berhasil menemukan solusi dan hasilnya diurai oleh sistem. Setiap baris dalam tabel schedules menautkan satu activity dengan day, time_slot, dan room yang telah dialokasikan. Dengan demikian, kualitas dan kelengkapan data dalam skema basis data ini secara langsung menentukan kualitas dan kelayakan solusi yang dapat ditemukan, karena basis data ini bukan hanya sekadar input, melainkan merupakan definisi masalah itu sendiri dalam format yang terstruktur.

Berikut adalah diagram entitas-hubungan yang menggambarkan struktur basis data FETNET.



Gambar 3 2 Diagram Entitas-Hubungan (ERD) Basis Data FETNET

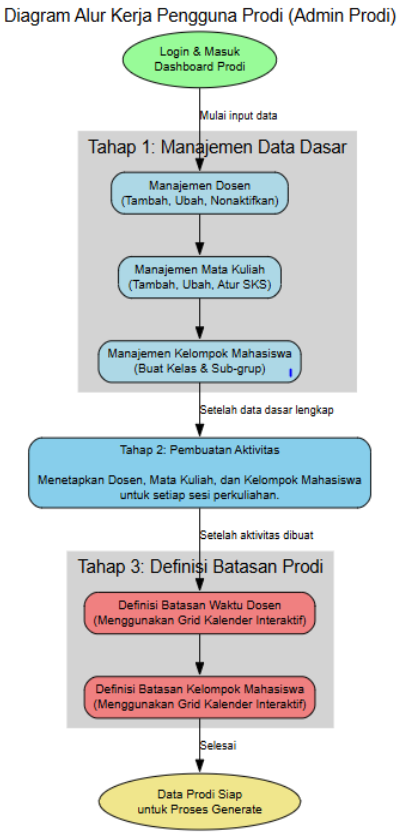
Diagram ini menggambarkan struktur logis dari basis data aplikasi FETNET. Setiap kotak mewakili sebuah entitas (tabel), dan garis yang menghubungkannya merepresentasikan relasi antar data, menyoroti peran sentral tabel activities dalam menghubungkan data master dan peran tabel schedules sebagai penyimpan hasil akhir.

3.5 Perancangan Antarmuka dan Alur Kerja Pengguna (UI/UX)

Perancangan UI/UX berfokus pada kejelasan, efisiensi, dan konsistensi, dengan alur kerja yang dirancang spesifik berdasarkan peran pengguna.

3.5.1 Alur Kerja Pengguna Prodi (Admin Prodi).

Admin Prodi bertanggung jawab atas pengelolaan data operasional harian yang spesifik untuk program studi mereka. Alur kerja untuk peran ini dirancang secara linear dan bertahap untuk memastikan kelengkapan dan akurasi data sebelum diserahkan ke tingkat fakultas. Sebagaimana divisualisasikan pada Gambar 3.3, alur kerja ini dimulai dengan Manajemen Data Dasar, di mana admin mengelola data inti seperti daftar dosen, mata kuliah, dan kelompok mahasiswa yang relevan untuk prodinya. Langkah selanjutnya adalah Pembuatan Aktivitas Perkuliahan, sebuah tahap krusial di mana data-data dasar tersebut dirangkai menjadi unit-unit yang siap dijadwalkan, dengan menetapkan dosen pengajar, mata kuliah, dan kelompok mahasiswa untuk setiap sesi perkuliahan. Tahap terakhir adalah Definisi Batasan Prodi, di mana admin menggunakan antarmuka visual seperti *grid* kalender interaktif untuk menandai slot-slot waktu di mana dosen atau kelompok mahasiswa di prodinya tidak bersedia atau tidak tersedia. Setelah ketiga tahap ini diselesaikan, data dari program studi tersebut dianggap lengkap dan siap untuk dikoordinasikan dalam proses penjadwalan tingkat fakultas. Berikut adalah diagram yang mengilustrasikan alur kerja pengguna Prodi.



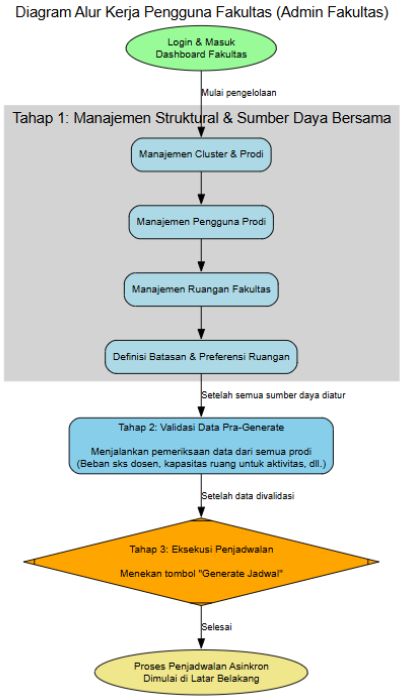
Gambar 3 3 Diagram Alur Kerja Pengguna Prodi

Diagram ini memvisualisasikan alur kerja linear yang dirancang untuk Admin Prodi, dimulai dari pengelolaan data master (dosen, mata kuliah, kelas), dilanjutkan dengan pembuatan aktivitas perkuliahan, dan diakhiri dengan definisi batasan waktu spesifik prodi.

3.5.2 Alur Kerja Pengguna Fakultas (Admin Fakultas)

Admin Fakultas bertindak sebagai koordinator, validator, dan manajer sistem secara keseluruhan. Alur kerja mereka, yang diilustrasikan pada Gambar 3.4, berfokus pada manajemen sumber daya bersama dan kontrol kualitas terpusat. Alur ini dimulai dengan Manajemen Sumber Daya Bersama, di mana Admin Fakultas mengelola data yang bersifat lintas-prodi, seperti inventaris ruangan, gedung, dan akun pengguna untuk setiap Admin Prodi. Tahap berikutnya yang sangat penting adalah Validasi Data Pra-Generate. Pada tahap ini, admin menjalankan fitur diagnostik sistem (*pre-flight check*) untuk mendeteksi secara otomatis adanya data

yang tidak lengkap, tidak konsisten, atau berpotensi menimbulkan konflik dari seluruh prodi yang telah diinput. Setelah memastikan semua data valid, Admin Fakultas memiliki wewenang tunggal untuk Eksekusi Penjadwalan, yaitu memicu proses *generate* jadwal untuk seluruh fakultas yang berjalan secara asinkron di latar belakang. Tahap akhir dari alur kerja ini adalah Visualisasi dan Distribusi Jadwal, di mana Admin Fakultas dapat melihat jadwal final yang telah berhasil dihasilkan oleh sistem, memverifikasinya, dan mendistribusikannya kepada semua pemangku kepentingan. Peran ini memastikan adanya kontrol kualitas dan koordinasi terpusat sebelum proses komputasi yang intensif dijalankan. Berikut adalah diagram yang mengilustrasikan alur kerja pengguna Fakultas.



Gambar 3 4 Diagram Alur Kerja Pengguna Fakultas

Diagram ini menggambarkan alur kerja manajerial untuk Admin Fakultas, yang mencakup pengelolaan sumber daya bersama, validasi data gabungan dari semua prodi, eksekusi proses penjadwalan, dan visualisasi hasil akhir.

3.6 Perancangan Pengujian Sistem

Pengujian sistem dirancang secara multi-level untuk memvalidasi aplikasi FETNET secara komprehensif.

3.6.1 Pengujian Integrasi

Pengujian integrasi berfokus pada verifikasi interaksi antara beberapa komponen sistem. Contoh skenario pengujian integrasi adalah memverifikasi alur lengkap dari pemicuan jadwal, pengiriman *job* ke antrian, eksekusi oleh *worker*, hingga pembaruan status di basis data dan antarmuka.

3.6.2 Pengujian Validasi dan Kelayakan (*User Acceptance Testing* - UAT)

Pengujian ini adalah tahap pengujian paling krusial yang mengukur keberhasilan sistem dalam memenuhi kebutuhan pengguna. Metode yang digunakan adalah Studi Kasus Implementasi Nyata dengan pendekatan *Black-Box Testing*. Instrumen utama dalam UAT ini adalah implementasi skala penuh untuk penjadwalan satu semester di FPTI UPI. Proses ini melibatkan migrasi data riil, definisi batasan kolaboratif, eksekusi iteratif, dan validasi silang oleh setiap prodi. Puncak dari pengujian ini adalah keberhasilan integrasi jadwal final yang dihasilkan oleh FETNET ke dalam sistem informasi akademik (SIAK) resmi universitas, yang memberikan bukti empiris paling kuat mengenai kelayakan dan efektivitas aplikasi.