BAB I
PENDAHULUAN

1.1 Latar Belakang

Manajemen proyek yang efisien merupakan bagian penting bagi operasional perusahaan berskala besar, tidak terkecuali bagi PT. GS Battery Semarang. Dalam praktiknya, kelancaran sebuah proyek sangat bergantung pada arus informasi antara pihak manajerial yang merencanakan tugas dan pihak operasional/teknis yang mengeksekusi tugas di lapangan. Namun, observasi di PT. GS Battery Semarang menemukan adanya kendala operasional yang disebabkan oleh penggunaan dua *platform* manajemen yang terpisah. Pihak manajemen menggunakan *platform OpenProject* untuk perencanaan strategis, sedangkan tim teknis di lapangan lebih memilih menggunakan *Trello* karena visualisasi *Kanban* yang interaktif. Kondisi ini memaksa staf administrasi untuk melakukan praktik *double entry* pada kedua sistem secara manual. Praktik ini tidak hanya menguras jam kerja produktif yang seharusnya dapat dialokasikan untuk tugas yang lebih strategis, tetapi juga menciptakan isolasi data antara pihak manajemen dan tim teknis.

Ketika pembaruan status pekerjaan di *Trello* tidak segera disalin ke *OpenProject*, muncul ketidakkonsistenan informasi yang mengakibatkan pihak manajerial kehilangan visibilitas proyek secara *real-time*. Hal ini sangat berisiko menghambat proses evaluasi dan pengambilan keputusan. Selain itu, tingginya intervensi manual sangat rentan terhadap kesalahan manusia, seperti tertinggalnya pembaruan tenggat waktu atau hilangnya detail instruksi kerja. Oleh karena itu, pembangunan sebuah sistem integrasi menjadi urgensi yang sangat mendesak. Integrasi ini dibutuhkan untuk menjembatani kedua *platform* secara otomatis, memastikan terciptanya satu sumber kebenaran data, dan membebaskan staf dari pekerjaan administratif yang repetitif.

Integrasi sistem antar *platform* ini sejalan dengan berbagai kajian yang telah dilakukan oleh para peneliti sebelumnya (dirangkum pada Tabel 2.1). Sebagai contoh, pemanfaatan arsitektur *middleware* dan *REST API* untuk menjembatani sistem informasi lintas *platform* telah dibuktikan keandalannya oleh Ramadhan dan Budiman (2020) serta Salim dan

Wahjono (2021). Selain itu, kajian komparatif mengenai mekanisme *webhook* dan *polling* dalam pertukaran data secara *real-time* untuk manajemen proyek juga telah dieksplorasi oleh Setiawan (2022). Lebih lanjut, Nugroho dan Santoso (2021) telah membuktikan bahwa implementasi *background service* sangat efektif untuk mengotomatisasi pemrosesan tugas *server*. Meskipun penelitian-penelitian tersebut memberikan landasan teori yang sangat esensial, sebagian besar solusi yang ditawarkan masih berfokus pada transfer data generik dan sinkronisasi satu arah. Penelitian terdahulu belum secara spesifik menangani permasalahan sinkronisasi dua arah dengan tingkat toleransi kegagalan jaringan yang tinggi, padahal kondisi ini sangat krusial dalam ekosistem perusahaan.

Untuk mengisi celah penelitian tersebut sekaligus menyelesaikan permasalahan spesifik di PT. GS Battery Semarang, penelitian ini difokuskan pada pengembangan sebuah sistem *middleware* integrasi berbasis *background service*. Sistem ini dibangun menggunakan *framework* .NET 8 dan bertindak sebagai penengah yang berinteraksi dengan antarmuka *REST API* dari *OpenProject* dan *Trello*. Cara kerja utamanya adalah dengan menarik atribut tugas yang baru dibuat atau diperbarui di *OpenProject* seperti Judul, Deskripsi, *Due Date*, dan Status untuk kemudian disinkronisasikan dan dipetakan secara otomatis ke dalam papan *Trello* tim teknis, begitu pula untuk pembaruan dari arah sebaliknya.

Dalam membangun *background service* yang andal, pustaka Hangfire dipilih sebagai mesin penjadwal utama. Pemilihan Hangfire didasari pada fakta bahwa proses integrasi yang bergantung pada API pihak ketiga sangat rentan terhadap gangguan latensi jaringan atau *downtime server* (Ramadhan & Budiman, 2020). Jika hanya menggunakan penjadwal standar, data yang gagal dikirim pada saat gangguan akan hilang. Hangfire memecahkan masalah ini dengan menyediakan arsitektur antrean persisten di dalam basis data SQL Server, sehingga antrean tugas tidak akan hilang meskipun server mengalami *restart*. Lebih dari itu, Hangfire memiliki fitur *Automatic Retry* dengan *Exponential Back-off* yang memungkinkan sistem melakukan pemulihan mandiri dan mencoba menyinkronkan data ulang secara otomatis ketika koneksi terputus (Oberauer, 2026). Keberadaan fitur ini menjadikan Hangfire sebagai teknologi yang paling ideal untuk menjamin konsistensi dan integritas data antara OpenProject dan Trello di PT. GS Battery Semarang.

Untuk menjamin keberhasilan implementasi di lapangan, aspek portabilitas dan *deployment* sistem juga menjadi perhatian utama dalam penelitian ini. Oleh karena itu, seluruh arsitektur *middleware* beserta basis data pendukungnya dibungkus menggunakan teknologi kontainerisasi *Docker*. Pemanfaatan *Docker* dan *Docker Compose* memastikan bahwa lingkungan eksekusi sistem termasuk dependensi .NET dan *SQL Server* terisolasi secara sempurna dan dapat berjalan konsisten tanpa terpengaruh oleh perbedaan spesifikasi atau konfigurasi *host* pada server lokal PT. GS Battery Semarang. Pendekatan kontainerisasi ini tidak hanya mengeliminasi potensi konflik dependensi, tetapi juga menyederhanakan proses instalasi dan pemeliharaan sistem secara keseluruhan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah bagaimana mengembangkan sistem integrasi yang mampu menghubungkan API dari *OpenProject* dengan *Trello* secara dua arah menggunakan *Hangfire* pada kasus di PT. GS Battery Semarang.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah merancang dan membangun sebuah sistem *middleware* integrasi berbasis *background service* yang mampu menghubungkan antarmuka API *OpenProject* dan *Trello* secara dua arah menggunakan pustaka *Hangfire* untuk mengatasi kendala *double entry* di PT. GS Battery Semarang.

1.4 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Bagi organisasi, sistem ini diharapkan dapat meningkatkan efisiensi operasional dengan menghilangkan beban administrasi input ganda dan menjamin konsistensi data proyek secara *real-time* atau terjadwal.
2. Bagi pengembang sistem, memberikan referensi teknis mengenai implementasi integrasi API pihak ketiga menggunakan pola arsitektur *background worker* dengan .NET dan *Hangfire*.
3. Bagi Akademisi, menambah pengetahuan terkait interoperabilitas sistem manajemen proyek dan implementasi *microservices* sederhana.

1.5 Batasan Masalah

Agar penelitian ini lebih terarah dan fokus, penulis menetapkan batasan masalah sebagai berikut:

1. Sinkronisasi data berfokus pada entitas utama tugas, yaitu: Judul (*Subject/Name*), Deskripsi, Tanggal Jatuh Tempo, dan Status/Posisi *List*.
2. Fitur sinkronisasi tidak mencakup lampiran *file* (*attachments*), komentar, atau *custom fields* yang kompleks.
3. Mekanisme sinkronisasi menggunakan metode *Polling* atau pengecekan berkala dengan interval waktu yang ditentukan, bukan *Real-time Webhook*.
4. Sistem diasumsikan berjalan pada jaringan yang stabil, dengan penanganan *error handling* terbatas pada *retry mechanism* bawaan *Hangfire* dan pencatatan log ke *database*.

1.6 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini disusun sebagai berikut:

1. BAB 1 PENDAHULUAN

   Bab ini menguraikan latar belakang permasalahan, perumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, serta sistematika penulisan laporan skripsi.
2. BAB 2 TINJAUAN PUSTAKA

   Bab ini memuat penjabaran mengenai teori-teori dasar, literatur, dan konsep yang menjadi landasan utama dalam pengembangan sistem. Pembahasan mencakup konsep manajemen proyek, arsitektur *middleware*, pengenalan *platform OpenProject* dan *Trello*, konsep *background service* menggunakan *Hangfire*, serta teori perancangan sistem menggunakan *Unified Modeling Language* (UML).
3. BAB 3 METODOLOGI PENELITIAN

   Bab ini menjelaskan tahapan dan pemodelan sistem yang akan dibangun. Fokus pembahasan meliputi analisis kebutuhan, perancangan arsitektur integrasi data, pemodelan interaksi sistem menggunakan diagram UML, hingga perancangan basis data.
4. BAB 4 HASIL DAN PEMBAHASAN

Bab ini berisi dokumentasi tahapan implementasi perancangan ke dalam bentuk kode program dan konfigurasi lingkungan *container Docker*. Selain itu, bab ini juga memaparkan hasil pengujian sistem secara komprehensif untuk memvalidasi fungsionalitas, keandalan sinkronisasi, dan kinerja *background service* yang telah dibangun.

5. BAB 5 PENUTUP

   Bab ini menyajikan kesimpulan yang ditarik dari hasil analisis, implementasi, dan pengujian sistem. Turut disertakan pula saran-saran konstruktif yang dapat digunakan sebagai acuan untuk pengembangan dan penyempurnaan sistem di masa mendatang.