Terakreditasi SINTA Peringkat 2
Surat Keputusan Direktur Jenderal Pendidikan Tinggi, Riset, dan Teknologi, Nomor: 158/E/KPT/2021
masa berlaku mulai Volume 5 Nomor 2 Tahun 2021 sampai Volume 10 Nomor 1 Tahun 2026

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 6 No. 1 (2022) 169 - 176 ISSN Media Elektronik: 2580-0760

Pengembangan Aplikasi Tiga-Tingkat Menggunakan Metode Scrum pada Aplikasi Presensi Karyawan Glints Academy

Zidni Iman Sholihati[1], Imam Tahyudin[2]
[1]Informatika, Fakultas Ilmu Komputer, Universitas Amikom Purwokerto
[2]Sistem Informasi, Fakultas Ilmu Komputer, Universitas Amikom Purwokerto
[1]zidni.imani@gmail.com, [2]imam.tahyudin@amikompurwokerto.ac.id

Abstract

The rapid development of technology requires a software development management system that can be adaptive in rapidly changing circumstances. Scrum is an agile method that has the advantage of being agile and adaptive. Glints Academy holds an Industry Project Exploration as the program to prepare students for the rapid development of technology and reduce the gap between the education field and industrial field by MBKM program from the Ministry of Education and Culture. This study aims to apply the Scrum method in a heterogeneous developer team and divergent ability backgrounds to build an application with three-level architecture. The developer team is college students who come from different regions spread across Indonesia with full online implementation. Scrum is used because it is advantageous to other methods in a relatively fast-changing environment and also provides good quality control. The sprints were carried out in two sprints with two weeks of development in each sprint. The application built is an employee attendance application with a three-tier architecture: client, server, and data. The client-tier application is a front-end server built using the React.js framework while the server-tier and data-tier are built-in back-end servers with the Node.js and Express.js frameworks. JWT (JSON Web Token) authentication determines access role to functions and resources available on the back-end server. The result is a web application that fulfills the entire product backlog determined by the product owner. The results of this research are this method can used to develop features enhancement in the middle of the application development process without affecting the main feature development and this method is effectively used for different team developer backgrounds and during its online development.

Keywords: scrum, software development management, three-tier architecture

Abstrak

Pesatnya perkembangan teknologi memerlukan sistem manajemen pengembangan perangkat lunak yang juga dapat beradaptasi dengan keadaan yang silih berganti. *Scrum* sebagai metode *agile* memiliki keunggulan sebagai metode yang selain tangkas juga dapat adaptif. Glints Academy yang mengadakan program *Industry Project Exploration* untuk mempersiapkan mahasiswa menghadapi pesatnya perkembangan teknologi dan mengurangi jurang antara dunia pendidikan tinggi dengan dunia kerja di industri sesuai dengan visi program MBKM dari Kemendikbud. Penelitian ini bertujuan untuk menerapkan metode *scrum* dalam lingkungan *developer team* yang heterogen dan latar belakang kemampuan divergen untuk membangun sebuah aplikasi dengan arsitektur tiga-tingkat. *Developer team* adalah mahasiswa yang berasal dari wilayah berbeda dan tersebar di Indonesia dengan pelaksanaan yang sepenuhnya berlangsung secara daring. *Scrum* digunakan karena lebih unggul dari metode lainnya dalam lingkungan industri yang relatif cepat berubah dan juga memberikan kontrol kualitas yang baik. *Sprint* dilaksanakan dalam dua kali *sprint* dengan tiap *sprint* dilaksanakan selama dua minggu. Aplikasi yang dibangun berupa aplikasi presensi karyawan dengan arsitektur tiga-tingkat yaitu sisi *client*, *server*, dan data. Sisi *client* aplikasi dalam bentuk *front-end server* dibangun menggunakan *framework* React.js sedangkan *sisi server* dan sisi data berada dalam *back-end server* dengan *framework* Node.js dan Express.js. Autentikasi dengan JWT (JSON *Web Token*) menentukan hak akses terhadap fungsi dan sumber daya yang tersedia dalam *back-end server*. Hasil yang diperoleh adalah sebuah aplikasi berbasis web yang telah memenuhi seluruh *product backlog* yang ditentukan oleh *product owner*. Hasil dari penelitian ini adalah metode yang diterapkan mampu mengembangkan fitur di tengah proses pengembangan aplikasi tanpa mempengaruhi pengembangan fitur utama dan metode ini efektif digunakan bagi lingkungan pengembang aplikasi yang berbeda yang pengembangannya berlangsung secara daring.

Kata kunci: *scrum*, manajemen pengembangan perangkat lunak, arsitektur tiga-tingkat.

Diterima Redaksi: 13-01-2022 | Selesai Revisi: 19-02-2022 | Diterbitkan Online: 27-02-2022

1. Pendahuluan

Laju perkembangan teknologi yang pesat mengharuskan keahlian adaptasi tinggi bagi orang yang terjun di dalamnya. Perkembangan yang kini menjadi pusat perhatian industri salah satunya adalah persiapan menghadapi industri 4.0. Glints sebagai perusahaan yang menumbuhkan ekosistem talenta *full-stack* dan *platform* rekrutmen di Asia Tenggara memiliki visi untuk mempersiapkan mahasiswa Indonesia menghadapi industri 4.0 dengan meluncurkan sebuah program *Industrial Project Exploration* (IPE) yang mendukung program MBKM (Merdeka Belajar Kampus Merdeka) Kemendikbud untuk mengurangi jurang antara dunia pendidikan tinggi dengan dunia kerja di industri atau dunia profesi nyata.

Program IPE berjalan secara daring penuh dengan kegiatan yang mengharuskan adanya kolaborasi dari mahasiswa yang tersebar dari wilayah di Indonesia dengan keahlian berbeda yang diarahkan untuk bekerja sama menerapkan keterampilan teknis masing-masing ke sebuah pengembangan aplikasi yang berasal dari mitra industri Glints.

Dari kondisi heterogen tersebut, posisi keahlian dapat dibagi dalam empat kategori yaitu *mobile developer*, *machine learning*/*data engineer*, *front-end developer*, dan *back-end developer*. Kategorisasi ini memudahkan pengembangan aplikasi dengan arsitektur tiga-tingkat sesuai dengan teknologi industri 4.0 yang terdiri dari lapisan tertentu dalam penerapan manufakturnya [1]. Arsitektur tiga-tingkat memberikan fleksibilitas terhadap *client* dan *server* yang digunakan sehingga memungkinkan penggunaan kecerdasan buatan yang memerlukan komputasi berat seperti *deep learning* yang terhubung dengan *mobile* [2] atau pembaharuan *framework* aplikasi untuk meningkatkan pengalaman pengguna [3]. Pemisahan *front-end server* sebagai sisi *client* dan *back-end server* sebagai sisi *server* juga dapat meningkatkan waktu respon sistem serta mengurangi beban *server* yang memberikan keunggulan bagi sistem *microservices* [4].

Pengembangan aplikasi dengan arsitektur yang terbagi-bagi serta keberagaman latar belakang mahasiswa sebagai pengembang aplikasi baik dari posisi keahlian ataupun kemampuan tentunya memerlukan metodologi yang tepat. Metode pengembangan aplikasi yang diterapkan harus berkarakter adaptif, tangkas, dan fleksibel. Keperluan karakteristik ini terdapat dalam *scrum* sebagai salah satu metode *agile* yang menekankan pada pengendalian perubahan yang mungkin terjadi selama pengembangan [5].

Penelitian sebelumnya mengenai efektivitas penerapan *scrum* dalam sektor bisnis menghasilkan efektivitas dalam dimensi *time*, *cost*, *quality* dan *scope* sehingga *scrum* dinilai mendukung perkembangan teknologi aplikasi yang pesat untuk memenuhi kebutuhan para penggunanya [6]. Beberapa perusahaan yang menerapkan *scrum* diteliti oleh Hayat dkk [7] menyimpulkan bahwa metode ini selain berdampak positif dalam manajemen perangkat lunak juga berdampak bagi manajemen sumber daya manusia. Hasil analisis penerapan metode ini terhadap penelitian sebelumnya juga menghasilkan kesimpulan bahwa metode ini lebih unggul dari metode lainnya dalam lingkungan industri yang relatif cepat dan juga memberikan kontrol kualitas yang baik karena terdapat pengujian fungsionalitas hasil di dalamnya [8].

Penelitian sebelumnya yang menggunakan metode serupa dalam lingkungan kepegawaian LIPI menghasilkan sebuah sistem informasi layanan kawasan dengan adanya efektifitas dan peningkatan fokus dalam kerja tim sehingga pekerjaan pengembangan sistem berlangsung lebih efisien. Hasil ini didapat dengan adanya penerapan prinsip *agile development* sesuai dengan konsep dan aturan *scrum* [9].

Berdasarkan penelitian sebelumnya yang diterapkan pada lingkungan kerja perbankan [6], sejumlah perusahaan pengembang aplikasi [7], dan sejumlah industri serta organisasi digital, penelitian ini memiliki ruang lingkup yang berbeda karena mahasiswa yang terlibat masih dalam proses belajar dan menjalankan program IPE dalam waktu singkat. Penelitian ini bertujuan untuk memaparkan hasil penerapan metode *scrum* dalam lingkungan mahasiswa sebagai pengembang aplikasi yang heterogen serta divergen dengan tujuan pengembangan aplikasi yang memiliki arsitektur tiga-tingkat. Aplikasi yang dibangun adalah aplikasi presensi karyawan yang merupakan permintaan dari *product owner* Glints.

Hasil yang ingin dicapai dari pengembangan aplikasi tiga-tingkat tersebut adalah adanya aplikasi sisi *client* dan *server* yang dapat berkomunikasi dengan baik sehingga memungkinan untuk digunakan oleh karyawan berdasarkan seluruh persyaratan yang disampaikan *product owner*.

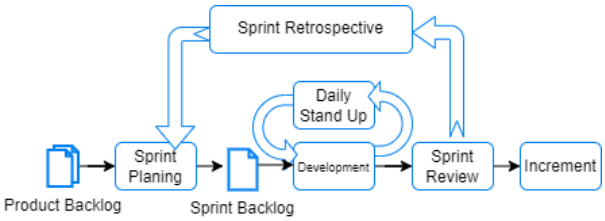
2. Metode Penelitian

Penelitian ini memiliki konsep *agile development* dengan menerapkan metode *scrum* yang memiliki beberapa peran: *product owner*, *scrum master*, dan *developer team*. *Product owner* adalah pemilik aplikasi yang bertanggung jawab dalam hasil kerja tim supaya memiliki nilai produk tinggi. *Scrum master* bertanggungjawab memimpin tiap tahap *scrum* dan memastikan setiap *sprint* berjalan lancar. Sementara *developer team* adalah sejumlah orang yang bekerja membangun sistem yang akan dirilis pada akhir *sprint*.

Artifak *scrum* terdiri dari *product backlog* dan *sprint backlog*. *Product backlog* adalah model informal dari pekerjaan yang perlu diselesaikan sebagai penjembatan

DOI: https://doi.org/10.29207/resti.v6i1.3793

antara konsep solusi dan pengembangan *software* [10]. *Product backlog* akan diturunkan menjadi *sprint backlog* yang berupa tugas kecil dari fungsi-fungsi aplikasi yang diberikan pada saat *sprint planning*.



Gambar 1. Metode *Scrum*

Tahapan *scrum* yang digambarkan pada Gambar 1 terdiri dari *sprint planning*, *daily stand up*, *sprint review* dan *sprint retrospective*. Terlaksananya seluruh tahapan ini dalam satu kali *sprint* menghasilkan *increment* yang merupakan rilis produk secara bertahap dari setiap *sprint* yang dilakukan [11].

2.1. *Sprint Planning*

Tahapan ini adalah tahap ketika *product owner* memberikan urutan prioritas terhadap *product backlog* yang telah dibuat di awal. *Product backlog* tersebut dikembangkan menjadi sebuah *sprint backlog* untuk dikerjakan *developer team* dalam setiap *sprint*.

Beberapa pertimbangan seperti tingkat kesulitan sebuah *product backlog*, teknologi yang akan digunakan, posisi keahlian (*mobile developer*, *machine learning*/*data engineer*, *front-end developer*, atau *back-end developer)*, dan ketentuan durasi iterasi *sprint* juga ditentukan dalam tahap ini.

2.2. Daily Stand Up

Tahap ini adalah pertemuan rutin yang bersifat singkat untuk seluruh anggota *developer team* dari *scrum master* guna memantau perkembangan proyek. Setiap *sprint backlog* yang telah dipindah ke fase diproses (*on going*) akan bersifat transparan dalam papan *scrum* yang dapat diakses seluruh anggota. Dalam rapat *daily stand up*, masing-masing tim pengembang melaporkan kemajuan pembuatan fitur, rencana tugas yang akan dilaksanakan hari itu, serta halangan (*blocker*) yang dihadapi.

Pertemuan hanya berlangsung selama 5 hingga 15 menit setiap hari aktif dan *scrum master* dapat mengarahkan hal yang diperlukan untuk sebuah *backlog* apabila terdapat *blocker* yang dapat menghalangi kelancaran proses pengembangan.

2.3. Sprint Review

Tahapan *sprint review* berisi penyampaian *product owner* mengenai *product backlog* yang telah dikerjakan selama satu *sprint*. Tahapan ini akan dihadiri lengkap oleh *product owner*, *scrum master*, dan *developer team*. *Scrum master* akan menginstruksikan *developer team* untuk mendemonstrasikan hasil pengerjaan dan melakukan pengujian-pengujian terhadap komponen yang telah dibuat. *Product owner* akan menguji serta memberi masukan pengembangan aplikasi sesuai dengan perencanaan produk. Tahapan ini menghasilkan sebuah peningkatan (*increment*) fitur produk yang selanjutnya akan dirilis.

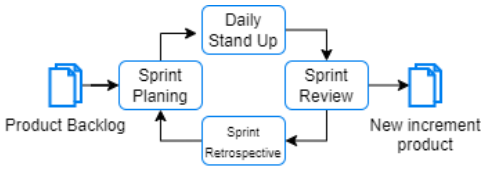
2.4. *Sprint Retrospective*

Tahapan ini adalah peninjauan *scrum master* terhadap hasil pekerjaan *developer team*. Hasil peninjauan ditujukan untuk memberi masukan agar kinerja dalam praktik *scrum* dapat berjalan lebih efektif dan dapat dipahami oleh setiap anggota tim pengembang.

Jika pengerjaan *product backlog* belum selesai atau ada penambahan dari *product owner*, maka akan ada penambahan iterasi *sprint* selanjutnya untuk memulai kembali *sprint planning*.

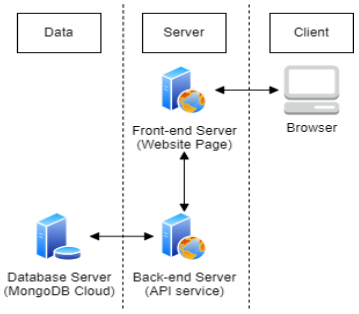
3. Hasil dan Pembahasan

3.1. Arsitektur



Gambar 2. Arsitektur Iterasi *Scrum*

Gambar 2 merupakan arsitektur alur iterasi yang terjadi dalam penerapan metode *scrum* penelitian ini. *Product backlog* sebagai daftar utama proses bisnis akan menjadi acuan pelaksanaan *sprint planning* yang kemudian dilanjutkan dalam tahap *daily stand up* sebagai tahap diskusi harian. Setelah berlangsung dua pekan atau 10 hari aktif, *scrum master* akan memimpin jalannya *sprint review* untuk melakukan demonstrasi aplikasi menjalankan seluruh pekerjaan yang telah selesai. Setelah *sprint review* menghasilkan sebuah peningkatan produk, *sprint retrospective* diadakan untuk mendiskusikan apa saja yang telah efektif, apa saja menjadi hambatan, apa yang dapat dilanjutkan dan yang tidak dapat dilanjutkan untuk *sprint* berikutnya.



Gambar 3. Arsitektur Sistem

DOI: https://doi.org/10.29207/resti.v6i1.3793

Arsitektur sistem penelitian yang direpresentasikan oleh Gambar 3 terdiri dari tiga tingkat yaitu *client, server,* dan data. *Client* sebagai tingkat *front-end server* melakukan presentasi tampilan ke pengguna. *Server* sebagai tingkat penengah *back-end server* yang menjadi pusat logika bisnis aplikasi. Tingkat data sebagai basis data menyimpan informasi dan sebagai pusat pengolahan dan manajemen data.

Front-end server dibangun menggunakan *framework* React.js sebagai salah satu *framework* yang menjalankan komputasi di dalam sisi *client* sehingga fungsi *routing* atau peralihan halaman tidak lagi dibebankan kepada *back-end server* [4]. *Back-end server* menggunakan Node.js dan *framework* Express.js yang memiliki kecepatan respon lebih baik dari beberapa bahasa pemrograman *server-side* yang umum digunakan untuk membangun sebuah API *service* [12]. Penyimpanan data dalam MongoDB *cloud* hanya dapat bisa diakses melalui *back-end server*. MongoDB sebagai basis data NoSQL dapat menangani permasalahan data yang tidak terstrukstur [13] yang sesuai dengan kemungkinan adanya pengembangan logika bisnis aplikasi dalam pengembangan metode *scrum*.

Arsitektur yang membagi sistem *back-end* dan *front-end* sebagaimana penelitian [14] memberikan keunggulan dalam permasalahan interoperabilitas sistem presensi yang menggunakan perangkat *fingerprint scanner*. Dalam penelitian ini, fungsi perangkat *fingerprint* digantikan dengan sistem deteksi lokasi karyawan dari kantor menggunakan GPS dalam perangkat gawai yang digunakan.

3.2. *Product Backlog*

Product backlog berisi daftar seluruh kebutuhan atau fitur aplikasi yang telah diurutkan berdasarkan prioritas yang selanjutnya akan digunakan sebagai sumber tiap perubahan atau pengembangan terhadap produk [15]. Dalam pengembangan aplikasi presensi karyawan ini, *product backlog* terbagi menjadi tiga domain: autentikasi, presensi, dan admin. Seluruh *product backlog* pada Tabel 1 didapatkan dari hasil penyampaian *product owner* dan setelah adanya diskusi bersama *scrum master*.

Tabel 1. *Product Backlog*

Domain	Deskripsi	Prioritas
Autentikasi	*User* bisa mendaftar menggunakan surel dan *password*	Tinggi
	User bisa *login* menggunakan surel dan *password*	Tinggi
	User bisa menggunakan fitur reset password menggunakan surel	Tinggi
	User bisa keluar dari aplikasi (*sign out*)	Sedang
Presensi	*User* bisa mengirim lokasi sekarang untuk melakukan presensi	Tinggi
	Jarak maksimal pengiriman lokasi yang diizinkan adalah 100 meter dari lokasi user dengan kantor	Tinggi
	Aplikasi bisa mengitung durasi kerja *user*	Sedang
Admin	Terdapat alur autentikasi bagi admin	Tinggi
	Terdapat halaman admin *dashboard* yang hanya diakses oleh admin	Tinggi
	Admin dapat menyetujui atau menolak pendaftaran akun	Tinggi
	Menampilkan user yang absen lebih dari tiga hari dalam sebulan	Sedang

3.2. *Sprint* 1

3.2.1. *Sprint Planning*

Dalam *sprint planning* pertama, *scrum master* memimpin rapat bersama *developer team* untuk menganalisis proses bisnis dari *product backlog*. *Developer team* dalam penelitian ini dikerucutkan menjadi dua tim saja yaitu tim *front-end* dan *back-end* karena *product backlog* yang ada tidak memerlukan posisi tambahan. Dengan begitu *front-end* mendapat penambahan anggota dari *mobile developer*. Serta *back-end* mendapatkan dari posisi *machine-learning/data engineer*. Iterasi *sprint* ditentukan sebanyak dua kali dengan tiap iterasi berjumlah dua minggu.

Hasil *sprint backlog* dibagi menjadi dua bagian yaitu *front-end* yang membangun tampilan antarmuka web dan *back-end* yang membangun API *service*. *Sprint backlog* disimpan dalam aplikasi Notion yang dapat diakses oleh seluruh tim. Pengembang terdiri dari tiga *front-end* dan *back-end* yang akan mengambil tugas dalam *sprint backlog* yang dapat dilihat pada Tabel 2 dan Tabel 3.

Tabel 2. *Sprint Backlog Front-End (Sprint 1)*

Deskripsi	Prioritas	Keterangan
Membuat desain UI	Tinggi	Merancang *wireframe* atau *interface* aplikasi
Membuat *layout* domain autentikasi	Tinggi	Membangun komponen tampilan halaman *register, login, forgot,* dan *reset password*
Membuat *layout* domain presensi	Tinggi	Membuat komponen tampilan untuk *check in, check out,* dan cek status presensi
Membuat *layout* domain admin	Tinggi	Membuat komponen daftar akun, persetujuan akun, dan cek absensi di halaman admin
Memasang alur logika komponen	Tinggi	Mengembangkan komponen agar untuk menjadi lebih interaktif sesuai logika program
Memasang integrasi API	Sedang	Menyambungkan komponen dengan layanan *back-end server* melalui integrasi API

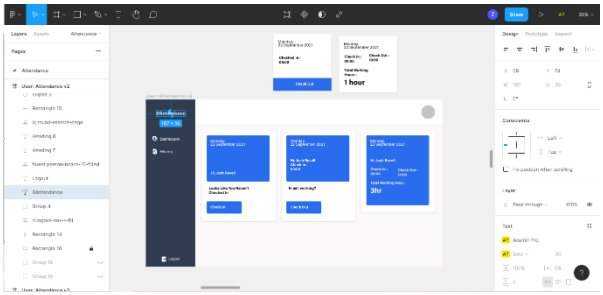
DOI: https://doi.org/10.29207/resti.v6i1.3793

Tabel 3. *Sprint Backlog Back-End (Sprint 1)*

Deskripsi	Prioritas	Keterangan
Membuat skema *database*	Tinggi	Menentukan skema tabel MongoDB
Membuat kerangka API *service*	Tinggi	Membuat struktur *controller-models-routes*
Membuat model tabel	Tinggi	Membuat konstruktor model dari skema tabel yang telah dibuat
Membuat *middleware*	Tinggi	Membuat fungsi yang digunakan untuk autentikasi dengan menggunakan JWT
Membuat API bagian autentikasi	Tinggi	Membuat API *register*, *login*, *forgot*, dan *reset* password
Membuat API bagian presensi	Tinggi	Membuat API *check in*, *check out*, dan cek status presensi
Membuat API bagian admin	Tinggi	Membuat API daftar akun, persetujuan akun, dan cek absensi
Membuat dokumentasi API	Sedang	Membuat dokumentasi menggunakan Swagger

3.2.2. *Daily Stand Up*

Daily stand up diadakan setiap hari aktif untuk memaparkan kemajuan pengembangan proyek dari *developer team* yang dipandu oleh *scrum master*. Pengembang *front-end* di tahap ini menyelesaikan tampilan UI seperti Gambar 4 yang dibangun menggunakan aplikasi Figma yang dapat digunakan secara kolaborasi. Gambar 4 merupakan rancangan desain halaman presensi yaitu halaman utama setelah *user* berhasil *login* lalu sistem akan menampilkan informasi *check in*, *check out*, dan *durasi kerja*.



Gambar 4. Tampilan UI Halaman Presensi

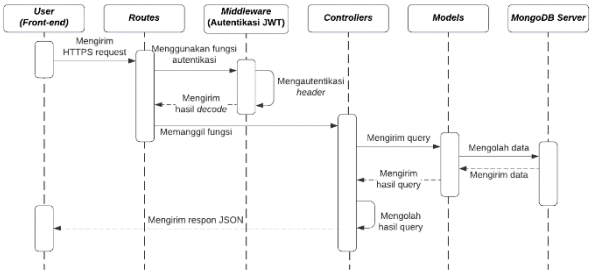
Dari pengembangan *back-end*, dapat dijelaskan hasil pembuatan skema tabel MongoDB dalam Tabel 4 yang berisi atribut dan tipe data dari tabel *attendances* dan tabel *accounts*. Kedua tabel dapat dihubungkan dengan agregasi dasar menggunakan atribut *account* di tabel *attendances* menuju tabel *accounts*. Agregasi dasar ini mengindikasikan salah satu kelas adalah bagian dari kelas lainnya.

Pengamanan layanan *back-end server* menggunakan autentikasi JWT untuk menentukan hak akses terhadap fungsi dan sumber daya yang tersedia. JWT (JSON *web token*) adalah sebuah protokol atau standar autentikasi yang memiliki sifat universal sehingga dapat digunakan dalam berbagai jenis bahasa pemrograman [16].

Tabel 4. Skema Database MongoDB

Tabel	Atribut	Tipe data
accounts	_id	objectId
	firstName	string
	lastName	string
	Surel	string
	password	string
	isAdmin	boolean
	status	string
	createdAt	timestamp
	updatedAt	timestamp
attendances	_id	objectId
	account	objectId
	inTime	timestamp
	inLocation	object (lat, long)
	outTime	timestamp
	outLocation	object (lat, long)
	createdAt	timestamp
	updatedAt	timestamp

Penggunaan JWT memungkinkan adanya keamanan dalam proses autentikasi, proses pengiriman dan respons, serta pengamanan data [17]. Komponen yang menangani hal ini adalah *middleware* yang dapat dipasang dalam alur logika program sebagaimana yang dijelaskan dalam *sequence diagram* Gambar 5.



Gambar 5. *Sequence Diagram* Program

Pembuatan dokumentasi API *service* ini menggunakan Swagger sebagai dokumentasi yang interaktif dan memungkinkan pengembang bagian *front-end* untuk berinteraksi secara langsung dengan layanan API. Swagger bersifat aplikatif, fungsional, dan solid untuk pengembangan API sehingga dapat digunakan dalam pengembangan ini. Alat pengembang API ini juga bisa membuat kode client dan server secara otomatis sehingga pengembangannya lebih efisien dan sederhana [18].

3.2.3. *Sprint Review*

Tahap ini berisi pemaparan *developer team* kepada *product owner* bersama *scrum master*. *Scrum master* menginstruksikan pengujian secara langsung dengan melakukan demonstrasi aplikasi. Hasil pengujian dalam Tabel 5 didasarkan pada *product backlog* sehingga demonstrasi dilakukan oleh bagian *front-end* yang telah mengintegrasikan komponen-komponen halaman web dengan API *service*.

DOI: https://doi.org/10.29207/resti.v6i1.3793

Tabel 5. Pengujian *Sprint Review* Pada *Sprint* 1

No	Deskripsi	Skenario	Hasil Uji
1	*User* bisa mendaftar menggunakan surel dan password	*User* mendaftar dengan surel dan password yang valid dan akan muncul informasi jika memasukkan data tidak valid	Gagal
2	*User* bisa *login* menggunakan surel dan *password*	*User* bisa *login* menggunakan surel dan *password* setelah diizinkan oleh admin	Berhasil
3	*User* bisa menggunakan fitur *reset password* menggunakan surel	*User* mendapatkan surel yang berisi halaman khusus untuk *reset password*	Berhasil
4	*User* bisa keluar dari aplikasi (*sign out*)	*User* bisa *logout* dari aplikasi dan menghapus sesinya	Berhasil
5	*User* bisa mengirim lokasi sekarang untuk melakukan presensi	Aplikasi dapat mendeteksi lokasi *user* dan mengirimkan datanya ke *server* ketika *check in/out*	Berhasil
6	Jarak maksimal pengiriman lokasi yang diizinkan adalah 100 meter dari lokasi user dengan kantor	*User* hanya dapat *check in/out* ketika jaraknya >= 100 meter berdasarkan lokasi GPS	Berhasil
7	Aplikasi bisa menghitung durasi kerja *user*	*User* dapat melihat lamanya waktu kerja setelah *check in*	Berhasil
8	Terdapat alur autentikasi bagi admin	Admin akan dialihkan ke halaman *dashboard* admin jika *login* menggunakan akun admin	Berhasil
9	Terdapat halaman admin *dashboard* yang hanya diakses oleh admin	Halaman *dashboard* admin tidak bisa diakses oleh non-admin	Berhasil
10	Admin dapat menyetujui atau menolak pendaftaran akun	Admin dapat mengubah status izin akun terdaftar menjadi *approred* atau *rejected* dan mengirimkan surel ke akun yang bersangkutan	Berhasil
11	Menampilkan *user* yang absen lebih dari tiga hari dalam sebulan	Admin dapat mendapatkan daftar karyawan yang tidak hadir sebanyak tiga hari dalam hitungan satu bulan di luar hari libur	Berhasil

Dalam pengujian pertama, halaman *register* tidak berhasil menampilkan informasi bagi *user* yang keliru memasukkan data-data ke dalam kolom formulir pendaftaran. Dari sebelas pengujian yang dilakukan, sepuluh berhasil dan satu gagal yang kemudian dicatat dan akan mendapatkan penanganan di *sprint* selanjutnya.

3.2.4. *Sprint Retrospective*

Tahap akhir dari *sprint* satu ini bertujuan untuk meninjau ke belakang seluruh hasil *developer team*. Terjadi diskusi antara peneliti dan sesama *developer team* mengenai kekurangan dan kelebihan dari hasil yang telah dikerjakan masing-masing. Umpan balik juga disampaikan untuk memaparkan apa yang sudah bagus dan apa yang masih perlu diperbaiki dari masing-masing individu. Pada akhir kegiatan, *scrum master* menyampaikan hasil kerja tim *back-end* yang sudah cepat sehingga memungkinkan adanya pengembangan API *service* bagian admin agar dapat menambahkan keterangan bagi karyawan yang tidak masuk tanpa informasi.

3.2. *Sprint* 2

3.2.1. *Sprint Planning*

Sprint backlog dalam *sprint* 2 didapatkan dari hasil *sprint review* dengan *scrum master* yang kembali menyusun *sprint backlog* dalam dua bagian. Fitur yang gagal dalam tahap pengujian akan mendapatkan kembali *sprint backlog* yang terkait untuk mendapatkan perbaikan *bug*.

Tabel 6. *Sprint Backlog Back-End (Sprint 2)*

Deskripsi	Prioritas	Keterangan
Membuat API koreksi presensi	Tinggi	Membuat fitur *update* presensi menambahkan keterangan bagi karyawan yang absen
Mengembangkan API daftar presensi	Tinggi	Mengembalikan daftar presensi dengan ikut mengembalikan daftar absensi untuk seluruh karyawan
Mengembangkan API daftar presensi satuan	Tinggi	Mengembalikan daftar presensi dengan ikut mengembalikan daftar absensi untuk seorang karyawan berdasarkan Id akun

Sprint backlog back-end dalam Tabel 6 berfokus pada pengembangan fitur koreksi presensi serta harus dipastikan hanya untuk presensi dari karyawan yang tidak masuk di hari kerja sehingga admin tidak bisa mengubah presensi yang telah direkam. Koreksi hanya memberikan keterangan atas alasan ketidakhadiran dan deskripsinya. Untuk mengembalikan daftar presensi yang tidak hanya memuat kehadiran, maka ditambahkan pengembangan API bagian admin untuk mengembalikan daftar absensi atau ketidakhadiran karyawan.

DOI: https://doi.org/10.29207/resti.v6i1.3793

Sprint backlog front-end dalam Tabel 7 yang mendapatkan kegagalan pengujian menambahkan *sprint backlog* untuk perbaikan *bug* di samping penambahan fitur koreksi presensi di dalam halaman *dashboard* admin.

Tabel 7. *Sprint Backlog Front-End (Sprint 2)*

Deskripsi	Prioritas	Keterangan
Mengembangkan fitur koreksi presensi	Tinggi	Membuat komponen fitur *update* presensi untuk menambahkan keterangan karyawan yang absen
Mengembangkan daftar tabel presensi seluruh karyawan	Tinggi	Menampilkan daftar presensi beserta daftar absensi untuk seluruh karyawan
Mengembangkan daftar tabel presensi satuan	Tinggi	Mengembalikan daftar presensi beserta daftar absensi untuk seorang karyawan berdasarkan Id akun
Memperbaiki *bug* fitur *register*	Sedang	Memperbaiki kesalahan *bug* di pemberian pesan yang tidak tampil pada halaman *register*

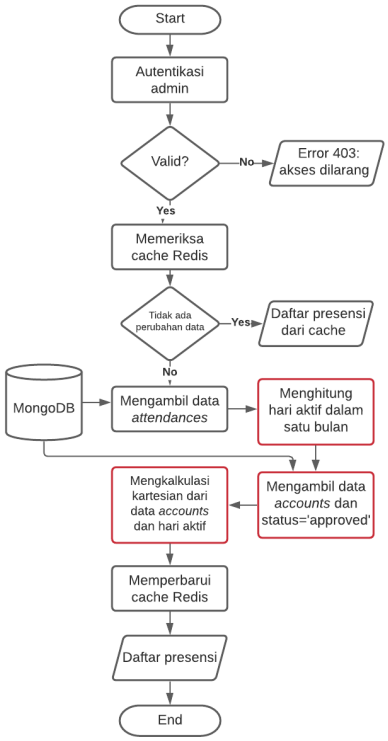
3.2.2. *Daily Stand Up*

Pengembangan fitur dalam *sprint backlog* kedua memerlukan pembaruan skema tabel dalam tabel *attendances*. Pembaruan yang dimaksud adalah menambahkan kolom *reason* dan *description* seperti yang dapat diamati dalam Tabel 8. Kolom *reason* berisi alasan ketidakhadiran karyawan seperti sakit, pergi, atau lainnya. Kolom *description* berisi keterangan dari alasan yang diberikan.

Tabel 8. Skema Tabel *Attendances* dalam *Sprint* 2

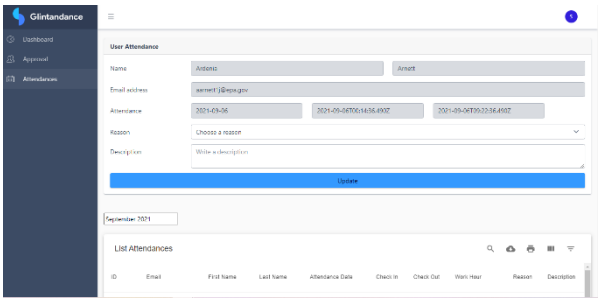
Atribut	Tipe data
_id	objectId
account	objectId
inTime	timestamp
inLocation	object (lat, long)
outTime	timestamp
outLocation	object (lat, long)
reason	string
description	string
createdAt	timestamp
updatedAt	timestamp

Selanjutnya adalah pengembangan API daftar presensi yang mengembalikan seluruh daftar presensi dan absensi. Dari alur algoritma awal yang hanya mengembalikan rekaman data kehadiran, alur algoritma kini diubah supaya dapat mengembalikan daftar absen tanpa penyimpanan pertama dalam tabel *attendances*. Alur algoritma yang direpresentasikan oleh Gambar 6 dalam pengembangan ini melakukan kalkulasi berdasarkan status akun yang terdaftar dan memiliki status diterima (*approved*).



Gambar 6. *Flow Chart* Pengembangan API Daftar Presensi

Bagian *front-end* menyesuaikan dengan penambahan kolom *reason* dan *description* dalam halaman admin seperti dalam Gambar 7. Adapun penanganan untuk menghalangi admin mengubah data karyawan dilakukan oleh bagian *back-end server* karena komputasi ini perlu berhubungan langsung dengan bagian basis data.



Gambar 7. Tampilan Pengembangan Halaman Dashboard Admin

3.2.3. *Sprint Review*

Developer team di tahap ini kembali melakukan demonstrasi hasil pengembangan produk kepada *product owner* dan *scrum master* untuk menilai hasil pengerjaan *sprint backlog* pada *sprint* 2 sesuai Tabel 9. Dalam tahapan ini, pengembangan produk telah berhasil diuji dan disetujui oleh *product owner* serta tidak ada perubahan atau penambahan kembali terhadap *sprint backlog*.

DOI: https://doi.org/10.29207/resti.v6i1.3793

Tabel 9. Pengujian *Sprint Review* Pada *Sprint* 2

No	Deskripsi	Skenario	Hasil Uji
1	*User* bisa mendaftar menggunakan surel dan *password*	*User* mendaftar dengan surel dan *password* yang valid dan akan muncul informasi jika memasukkan data tidak valid	Berhasil
2	Membuat API koreksi presensi	Admin dapat menambahkan keterangan dari ketidakhadiran karyawan	Berhasil
3	Mengembangkan API daftar presensi	Admin dapat melihat daftar presensi beserta absensi seluruh karyawan	Berhasil
4	Mengembangkan API daftar presensi satuan	Admin dapat melihat daftar presensi beserta absensi salah satu karyawan berdasarkan nomor Id karyawan	Berhasil

4. Kesimpulan

Berdasarkan uraian yang telah dipaparkan dapat diambil kesimpulan bahwa pengembangan aplikasi dengan arsitektur tiga-tingkat berhasil dibangun dan telah memenuhi seluruh ketentuan *product owner*. Penggunaan metode *scrum* memungkinan adanya pengembangan fitur di tengah proses pengembangan aplikasi tanpa mempengaruhi pengembangan fitur utama. Metode *scrum* efektif digunakan bagi lingkungan pengembang aplikasi yang masih dalam tahap belajar seperti mahasiswa yang memiliki latar belakang berbeda dan selama pengembangannya berlangsung secara daring sepenuhnya.

Adapun saran berdasarkan hasil penelitian ini adalah perlunya penelitian lanjutan yang menganalisa penyebab efektifitas metode *scrum* dalam lingkungan yang heterogen dan divergen ini. Penelitian selanjutnya juga dapat mengembangkan arsitektur tiga-tingkat sisi *client* yang berbasis *mobile* untuk memberikan pengalaman terbaik bagi *user* juga sebagai implementasi penuh kelebihan arsitektur ini dalam sisi fleksibilitasnya.

Ucapan Terimakasih

Kepada Universitas Amikom Purwokerto yang telah membantu dalam penerbitan artikel penelitian ini.

Daftar Rujukan

[1] R. Perumalraja, B. P. Nivetha, G. Priyanka, and P. Sowmiya, "Novel Three-Tier Architecture for Implementing Industry 4.0," *Int. Res. J. Eng. Technol.*, vol. 07, no. 04, pp. 4388–4393, 2020.

[2] Miftakhurrokhmat, R. A. Rajagede, and R. Rahmadi, "Presensi Kelas Berbasis Pola Wajah, Senyum dan Wi-Fi Terdekat dengan Deep Learning," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 5, no. 1, pp. 31–38, 2021, doi: 10.29207/resti.v5i1.2575.

[3] Tofid, E. Julianto, and Y. Harjoseputro, "Jurnal Resti," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 5, pp. 923–925, 2020, doi: 10.29207/resti.v4i5.2325.

[4] Y. Gong, F. Gu, K. Chen, and F. Wang, "The Architecture of Micro-services and the Separation of Frond-end and Back-end Applied in a Campus Information System," *Proc. 2020 IEEE Int. Conf. Adv. Electr. Eng. Comput. Appl. AEECA 2020*, pp. 321–324, 2020, doi: 10.1109/AEECA49918.2020.9213662.

[5] M. Morandini, T. A. Coleti, E. Oliveira, and P. L. P. Corrêa, "Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams," *Comput. Sci. Rev.*, vol. 39, p. 100314, 2021, doi: 10.1016/j.cosrev.2020.100314.

[6] Shandy, "Efektivitas Scrum Pada Manajemen Proyek Teknologi Informasi Di Pt Bank Central Asia Tbk.," *J. Manaj. Bisnis dan Kewirausahaan*, vol. 3, no. 4, pp. 32–40, 2019, doi: 10.24912/jmbk.v3i4.4989.

[7] F. Hayat, A. U. Rehman, K. S. Arif, K. Wahab, and M. Abbas, "The Influence of Agile Methodology (Scrum) on Software Project Management," *Proc. - 20th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2019*, no. July, pp. 145–149, 2019, doi: 10.1109/SNPD.2019.8935813.

[8] H. R. Suharno, N. Gunantara, and M. Sudarma, "Analisis Penerapan Metode Scrum Pada Sistem Informasi Manajemen Proyek Dalam Industri & Organisasi Digital," *Maj. Ilm. Teknol. Elektro*, vol. 19, no. 2, p. 203, Dec. 2020, doi: 10.24843/MITE.2020.v19i02.P12.

[9] W. Warkim, M. H. Muslim, F. Harvianto, and S. Utama, "Penerapan Metode SCRUM dalam Pengembangan Sistem Informasi Layanan Kawasan," *J. Tek. Inform. dan Sist. Inf.*, vol. 6, no. 2, pp. 365–378, 2020, doi: 10.28932/jutisi.v6i2.2711.

[10] T. Sedano, P. Ralph, and C. Peraire, "The Product Backlog," *Int. Conf. Softw. Eng.*, pp. 200–211, 2019, doi: 10.1109/ICSE.2019.00036.

[11] R. I. Pratama, "Pengembangan Back End Bagian Provider Pada Marketplace Travinesia.com Dengan REST API," Institut Pertanian Bogor, 2018.

[12] A. C. Rompis and R. F. Aji, "Perbandingan Performa Kinerja Node.js, PHP, dan Python dalam Aplikasi REST," *Cogito Smart J.*, vol. 4, no. 1, p. 160, 2018, doi: 10.31154/cogito.v4i1.92.160-170.

[13] K. Srivastava, D. Kumar Choubey, and J. Kumar, "Implementation of Inventory Management System," *SSRN Electron. J.*, 2020, doi: 10.2139/ssrn.3563375.

[14] A. Rahmatulloh, R. Gunawan, and I. Darmawan, "Web Services to Overcome Interoperability in Fingerprint-based Attendance System," *Atl. Highlights Eng.*, vol. 2, no. IcoIESE 2018, pp. 277–282, 2019, doi: 10.2991/icoiese-18.2019.49.

[15] R. Wirfs-Brock and L. B. hvatum, "Even More Patterns for the Magic Backlog," *Proc. 25th Conf. Pattern Lang. Programs*, pp. 1–17, 2018.

[16] A. A. Ridha, H. Ajie, and M. F. Duskarnaen, "Pengembangan Web Service Sistem Pembayaran Multibank Universitas Negeri Jakarta," *PINTER J. Pendidik. Tek. Inform. dan Komput.*, vol. 5, no. 1, pp. 25–33, 2021, doi: 10.21009/pinter.5.1.4.

[17] A. Umarjati and A. Wibowo, "Implementasi JWT pada Aplikasi Presensi dengan Validasi Fingerprint ," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 1, no. 10, pp. 1085–1091, 2021, doi: 10.29207/resti.v4i6.2650.

[18] J. Larsson and L. Åkermark, "The value of implementing API-first methodology when developing APIs," Jönköping University, 2021.

DOI: https://doi.org/10.29207/resti.v6i1.3793