BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian dengan judul, "Rancang Bangun *Back-end API* pada Aplikasi *Mobile* AyamHub Menggunakan *Framework Node JS Express* [11]" membahas tentang pengembangan *REST API* untuk merancang suatu *software* berbasis *mobile* bernama AyamHub yang digunakan untuk menghubungkan sistem antara peternakan dan UMKM di Indonesia. Penelitian ini menerapkan *REST API* dengan menggunakan *framework Express JS* yang berbasis *Node JS* dan *Sequalize* sebagai *database* relasional, serta metode *Waterfall* untuk pengembangannya. Penelitian ini menghasilkan implementasi dari perancangan *API* dan menerapkannya untuk aplikasi *mobile* dengan pengujian *black box* yang menunjukkan semua *endpoint* berjalan dengan status sukses.

Penelitian dengan judul, "Sistem Informasi Pemesanan Tiket Wisata Alam Berbasis *Website* di Taman Nasional Baluran dengan *PHP* & *MySQL* [1]" membahas tentang penggunaan *softeare* untuk pemesanan tiket pada Taman Nasional Baluran yang dibangun berbasis *website* dengan menerapkan *REST API*. Penelitian ini menerapkan *REST API* dengan menggunakan bahasa pemrograman *PHP* dan *database MySQL,* serta metode *Waterfall* sebagai metode pengembangannya. Hasil dari penelitian ini berupa *website* yang mencakup layanan pemesanan tiket, pembayaran, validasi dengan pemindaian *barcode,* dan juga menyediakan informasi mengenai Taman Nasional Baluran.

Penelitian dengan judul, "Implementasi *REST* Dalam Membangun *Web Service* Menggunakan *Golang* (Studi Kasus: Aplikasi Satudikti) [12]" membahas tentang pengembangan teknologi *REST* dengan *Web Service* pada aplikasi Satudikti dengan ruang lingkupnya adalah membuat *API Contract*, *Database*, dan *REST API*. Bahasa pemrograman yang digunakan pada penelitian ini adalah *Golang* dengan *framework Echo* untuk implementasi *REST API* dan metode pengembangan *Scrum*. Hasil dari penelitian ini berupa *endpoint* dengan protokol *HTTP* dalam arsitektur *REST* yang terdiri dari method *GET, POST, PUT,* dan *DELETE*

Penelitian dengan judul, "Pengembangan *REST API* SIABANG (Sistem Administrasi Pembangunan) Menggunakan *Java* [13]" membahas tentang pembuatan *REST API* untuk Sistem Informasi Administrasi Pembangunan dengan *framework Spring*. Hasil dari penelitian ini yaitu didapatkan bahwa untuk perancangan *REST API* untuk layanan pertukaran data telah berhasil dan ketika ada perubahan data, *API* akan mengirimkan data yang diperbarui. Setelah itu, dilakukan pengujian menggunakan *Postman* untuk *API* yang telah dirancang.

Penelitian dengan judul, "Perancangan *Microservice* Berbasis *REST API* pada *Google Cloud Platform* Menggunakan *NodeJS* dan *Python* [14]" membahas tentang pengembangan sistem dengan metode *Extreme Programming* dan *REST API* dengan *framework Express JS* yang menghasilkan *endpoint* dari fitur-fitur seperti *Authentication*, Hasil panen, Inventaris, Keuangan, dan Penyakit dengan pengujian menggunakan metode *black box*. Pada uji fungsionalitas menunjukkan bahwa *performance* sistem yang dikembangkan memiliki *average* 40,07 *hits*/detik dengan *average* waktu responsnya adalah 60,41*ms*.

Penelitian dengan judul, "Rancang Bangun Aplikasi Penjualan *Online* Berbasis *Web* Menggunakan Metode *Scrum* [15]" membahas tentang permasalahan yang berkaitan dengan sistem penjualan yang masih menggunakan cara manual dan belum adanya wadah untuk memasarkan produk secara *online* dengan fitur-fitur seperti cetak laporan penjualan, rekap laporan penjualan, dan promosi melalui media sosial. Penelitian ini menghasilkan halaman *admin* aplikasi penjualan *online* yang dapat mengelola pesanan dan dilakukan pengujian sistem yang menunjukkan hasil sukses. Penggunaan metode *scrum* di sini membantu kepentingan pemangku dalam penyesuaian kebutuhan sistem.

Tabel 2.1 Penelitian Terdahulu

Judul	Masalah	Metode	Hasil	Kekurangan Penelitian	Perbedaan
Rancang Bangun *Back-end API* pada Aplikasi *Mobile* AyamHub Menggunakan *Framework Node JS Express* [11]	Belum adanya pengembangan *back-end* pada *software mobile* AyamHub yang memudahkan penjualan antara penjual ayam boiler dengan peternakan. Penelitian ini membahas tentang perancangan *back-end* dengan arsitektur *REST API* pada aplikasi AyamHub.	*Waterfall*	Penelitian ini menghasilkan *endpoint REST API* yang digunakan oleh aplikasi *mobile* AyamHub sebagai implementasi dari *REST API* dengan pengujian *black box testing* yang menujukkan hasil semua *API* dapat berjalan lancar (statusnya sukses).	Pengembangan *REST API* yang digunakan masih menggunakan *API* dasar seperti untuk *register, login, updateData*, tetapi belum adanya *API* untuk lupa kata sandi, hapus akun, *logout*, dan *API* penunjang lainnya.	Pada penelitian terdahulu menggunakan *framework ExpressJS* dalam mengembangkan *REST API*. Pada penelitian ini menggunakan *framework Ruby on Rails* dalam mengembangkan *REST API*. Pada penelitian terdahulu dengan metode *Waterfall*. Pada penelitian ini menggunakan metode *Scrum.*
Sistem Informasi Pemesanan Tiket Wisata Alam Berbasis *Website* di Taman Nasional Baluran dengan *PHP* & *MySQL* [1]	Belum ada sistem informasi pemesanan tiket secara *online* di Taman Nasional Baluran dan masih menggunakan pemesanan tiket manual.	*Waterfall*	Hasil penelitian berupa sistem informasi Taman Nasional Baluran dengan tampilan *interface register user*, *login user*, halaman *user*, halaman pemesanan tiket, tampilan *booking* tiket, dan tampilan e-tiket.	Tidak diberikan hasil pengujian terhadap aplikasi yang telah dibangun.	Pada penelitian terdahulu menggunakan bahasa pemrograman *PHP* dan *database MySQL* dalam mengembangkan *REST API*. Pada penelitian ini menggunakan bahasa pemrograman *Ruby* dan *database PostgreSQL dalam* mengembangkan *REST API*. Pada penelitian terdahulu menggunakan metode

Judul	Masalah	Metode	Hasil	Kekurangan Penelitian	Perbedaan
					Waterfall, metode ini meliputi pengumpulan data, studi pustaka, perencanaan sistem, implementasi, uji coba, dan dokumentasi. Pada penelitian ini menggunakan metode *Scrum* yang meliputi *Product Backlog*, *Sprint Planning*, *Sprint Backlog*, *Daily Scrum*, *Sprint Review*, dan *Sprint Retrospective.*
Implementasi *REST* Dalam Membangun *Web Service* Menggunakan *Golang* (Studi Kasus: Aplikasi Satudikti) [12]	Diperlukan metode untuk membangun *web service* pada aplikasi Satudikti, metode tersebut adalah *REST*.	*Scrum*	Hasil dari penelitian ini berupa *endpoint* dengan protokol *HTTP* dalam arsitektur *REST* yang terdiri dari *method GET, POST, PUT,* dan *DELETE* untuk aplikasi Satudikti.	Penelitian ini pengujiannya menggunakan *Postman* dan tidak diketahui metode pengujiannya.	Pada penelitian terdahulu menggunakan *framework Echo* dalam mengembangkan *REST API*. Pada penelitian ini menggunakan *framework Ruby on Rails* dalam mengembangkan *REST API*. Penelitian terdahulu menggunakan metode *scrum* selama empat bulan untuk pengerjaan proyek yang meliputi *Sprint Refinement, Daily Scrum, Technical*

Judul	Masalah	Metode	Hasil	Kekurangan Penelitian	Perbedaan
					Meeting, Sprint Review, dan Koordinasi Proyek.
Pengembangan *REST API* Siabang (Sistem Administrasi Pembangunan) Menggunakan *Java* [13]	Belum adanya fitur untuk mengolah informasi, penelitian ini merancang Tabel Urusan pada *back-end* untuk memenuhi kebutuhan pada aplikasi Siabang.	*Extreme Programming* (*XP*)	Hasil dari penelitian ini berupa perancangan *REST API* dengan *framework Spring* dengan membuat *model, service,* dan *controller* pada aplikasi Siabang serta dilakukan pengujian menggunakan *Postman* untuk *API* yang telah dirancang.	Tidak diketahui metode pengujiannya.	Pada penelitian terdahulu menggunakan bahasa pemrograman *Java* dalam mengembangkan *REST API*. Pada penelitian ini menggunakan bahasa pemrograman *Ruby dalam* mengembangkan *REST API*. Pada penelitian terdahulu menggunakan metode *Extreme Programming (XP)*. Pada penelitian ini menggunakan metode *Scrum*.
Perancangan *Microservice* Berbasis *REST API* pada *Google Cloud Platform* Menggunakan *NodeJS* dan *Python* [14]	Belum adanya arsitektur *microservice* dalam pengembangan *REST API* pada aplikasi RIFSA.	*Extreme Programming* (*XP*)	Hasil dari penelitian ini berupa *endpoint* dari fitur-fitur *CRUD* seperti *Authentication,* Hasil panen, Inventaris, Keuangan, dan Penyakit, lalu dilakukan pengujian dengan metode *Black box.* Pada uji fungsionalitas menunjukkan bahwa *performance* sistem yang	Tidak diberikan hasil pengujian dengan metode *black box,* hanya memberikan hasil *performance test*.	Pada penelitian terdahulu menggunakan bahasa pemrograman *NodeJS* dan *Python* dalam mengembangkan *REST API*. Pada penelitian ini menggunakan bahasa pemrograman *Ruby dalam* mengembangkan *REST API*. Pada penelitian terdahulu menggunakan metode

Judul	Masalah	Metode	Hasil	Kekurangan Penelitian	Perbedaan
			dikembangkan memiliki *average* 40,07 *hits*/detik dengan *average* waktu responsnya adalah 60,41ms.		*Extreme Programming*. Pada penelitian ini menggunakan metode *Scrum.*
Rancang Bangun Aplikasi Penjualan *Online* Berbasis *Web* Menggunakan Metode *Scrum* [15]	Belum adanya wadah untuk memasarkan produk secara *online* dengan fitur-fitur seperti cetak laporan penjualan, rekap laporan penjualan, dan promosi melalui media sosial.	*Scrum*	Hasil dari penelitian ini adalah halaman *admin* aplikasi penjualan *online* yang dapat mengelola pesanan dan dilakukan pengujian sistem yang menunjukkan hasil sukses. Penggunaan metode *scrum* di sini membantu kepentingan pemangku dalam penyesuaian kebutuhan sistem.	Peneliti tidak menyertakan metode pengujian yang digunakan pada pengembangan aplikasi ini.	Pada penelitian terdahulu dengan penerapan metode *scrum* terdapat *product backlog* yang berisi kebutuhan aplikasi, *sprint* yang berisi *sprint planning* dan *sprint backlog*, *daily scrum* untuk memantau perkembangan setiap harinya, dan *sprint review* ketika serangkaian *sprint* telah dikerjakan.

2.2 Landasan Teori

2.2.1 Pariwisata

Pariwisata adalah salah satu pilar penting dalam pembangunan berkelanjutan yang berpotensi untuk meningkatkan pendapatan negara, menciptakan lapangan pekerjaan, dan berperan dalam menambah devisa negara. Berdasarkan potensi tersebut, pemerintah daerah dapat meningkatkan pembangunan sarana dan prasarana di sektor ini untuk meningkatkan daya tarik wisatawan [16]. Kepariwisataan di Indonesia telah diatur dalam Undang-Undang nomor 10 Tahun 2009. Pariwisata menurut Undang-Undang Republik Indonesia Nomor 10 Tahun 2009 merupakan berbagai macam kegiatan wisata dan didukung berbagai fasilitas serta layanan yang disediakan oleh masyarakat, pengusaha, Pemerintah, dan Pemerintah Daerah [17]. Pengembangan sektor pariwisata tidak hanya menambah pendapatan negara, tetapi juga memberikan manfaat langsung bagi masyarakat dengan keterlibatan mereka dalam kegiatan pariwisata dan memberikan hubungan yang baik antara masyarakat dan pariwisata [18].

2.2.2 Curug Pletuk

Curug Pletuk merupakan suatu objek wisata air terjun yang berada pada Desa Pesangkalan, Kecamatan Kecamatan Pagedongan, Kabupaten Banjarnegara. Curug Pletuk memiliki keindahan alam yang indah dan masih terjaga alamnya dengan pemandangan air terjun yang jernih disertai udara sekitar yang masih sejuk [2].

2.2.3 *Website*

Website merupakan rangkaian halaman informasi yang disertai informasi pendukung, seperti gambar, video, dan data lainnya yang disimpan pada suatu *domain* dan dapat diakses melalui internet [19]. *Website* terdiri dari dua jenis yaitu sebagai berikut [20].

1. *Website* Statis

Website Statis merupakan tipe *website* yang tidak dapat diberikan perubahan oleh pengguna. Pengguna tidak dapat melakukan pengubahan isi

konten dari suatu *website* dengan mudah, kecuali pengguna mempunyai akses untuk masuk ke dalam struktur kode dan *database* sistem.

2. *Website* Dinamis

Website Dinamis merupakan tipe *website* yang fleksibel dalam perubahan dengan menyesuaikan konten dengan sendirinya tanpa membutuhkan perubahan manual pada struktur kode.

2.2.4 *Backend Development*

Backend Development adalah tahap pengembangan di sisi *server* yang berfokus pada pengolahan *database*, *scripting*, dan struktur dasar sebuah *website* [21]. *Backend* tidak melakukan interaksi dengan pengguna secara langsung. *Backend* menyediakan data melalui *server* dan digunakan oleh aplikasi *client* untuk pengaksesan data melalui *Application Programming Interface* (*API*) [22]. *API* merupakan antarmuka yang digunakan pengembang program untuk mengakses layanan atau aplikasi, serta dapat menggunakan fungsi yang sudah ada dari aplikasi lain dengan menggunakan *API* tanpa perlu membangun dari awal [23]. Bahasa pemrograman yang dapat digunakan dalam *backend development* meliputi *JavaScript*, *PHP*, *Ruby*, *Python*, dan sebagainya. Pengembangan *backend* dibangun dengan menggunakan kode pemrograman untuk mengatur dan menyediakan data yang akan ditampilkan pada antarmuka pengguna (UI) di *frontend* [24]. Salah satu pengembangan *Backend Development* untuk meningkatkan fungsionalitas sistem adalah dengan merancang *REST API*.

2.2.5 *REST API*

Representational State Transfer Application Programming Interface (REST API) merupakan pengembangan dari *API* yang menerapkan arsitektur *REST* dengan tujuan untuk mempermudah proses transmisi data dalam suatu jaringan [25]. Transmisi data dilakukan dengan melalui *interface* yang telah terstandarisasi, yaitu protokol *HTTP* [4]. Arsitektur *REST* memiliki beberapa komponen adalah sebagai berikut [26].

1. URI Design

Protokol *HTTP* digunakan oleh *REST API* untuk pengaksesan sumber daya pada *API* dengan memerlukan *Uniform Resource Identifier (URI)* yang disebut *endpoint.* Contoh *endpoint* yaitu *products, products/1*, dan lain-lain.

2. HTTP Verbs

HTTP Verbs merupakan metode untuk mengetahui keinginan dari klien dengan mengirim request ke *server*. Metode yang digunakan yaitu seperti *GET, POST, PUT,* dan *DELETE*.

3. HTTP Response Code

HTTP Response *Code* merujuk pada kode yang telah menjadi standar dalam protokol *HTTP* saat klien melakukan permintaan. Terdapat tiga jenis kode yang umumnya digunakan dalam *REST API*, yaitu:

- Kode 2XX, menunjukkan bahwa *request* yang diajukan oleh klien berhasil.
- Kode 4XX, menunjukkan bahwa terdapat *error* pada klien ketika mengirim *request*.
- Kode 5XX, menunjukkan bahwa terdapat *error* pada *server* ketika mengirim *request*.

4. Format Response

Setiap kali klien melakukan permintaan, *server* akan memberikan *response* yang berisi data atau informasi. *REST API* biasanya menggunakan dua format respons seperti *Extensible Markup Language* (*XML*) dan *JavaScript Object Notation* (*JSON*).

2.2.6 *Ruby*

Ruby merupakan sebuah bahasa pemrograman *open-source* yang bersifat dinamis, mudah dipahami, dan mendukung produktivitas yang tinggi [27]. *Ruby* bertujuan untuk mengintegrasikan keunggulan dari seluruh bahasa pemrograman *script* [28]. *Ruby* dikembangkan dengan bahasa pemrograman *C* dengan fitur dasar yang mirip dengan *Perl* dan *Python* [29]. Salah satu pengembangan dalam penggunaan bahasa pemrograman *Ruby* adalah dengan menggunakan *framework Ruby on Rails* untuk merancang suatu *REST API*.

2.2.7 *Ruby on Rails*

Ruby on Rails adalah *framework* pengembangan aplikasi web yang dibangun menggunakan bahasa pemrograman *Ruby* yang mencakup seluruh komponen yang diperlukan untuk pengembangan aplikasi web dan mendukung basis data sesuai dengan konsep *Model-View-Controller (MVC)* [30]. Konsep *Model-View-Controller (MVC)* pada *Ruby on Rails* diterapkan pada *design-pattern* dengan mengolaborasikan beberapa komponen untuk menampilkan dan mengolah informasi yang diminta maupun dikirim oleh klien [31]. *Ruby on Rails* memiliki beberapa keunggulan sebagai berikut [32].

1. Metaprogramming

Ruby on Rails menggunakan teknik *metaprogramming*, yang memungkinkan penggunaan program untuk menulis program. Ini memungkinkan pengguna untuk lebih produktif dan menggantikan teknik generasi kode yang rumit.

2. Active Record

Ruby on Rails memperkenalkan kerangka kerja *Active Record* yang memungkinkan penyimpanan objek ke dalam *database* dengan cara yang sederhana dan efisien. Ini memanfaatkan *metaprogramming* untuk menghubungkan objek *domain* dengan skema *database*.

3. Convention over Configuration

Ruby on Rails mengutamakan konvensi daripada konfigurasi. Ini berarti pengembang tidak perlu menulis banyak kode konfigurasi jika mereka mengikuti konvensi umum. Hal ini dapat mengurangi jumlah kode konfigurasi hingga lima kali lipat lebih sedikit daripada *framework Java* yang serupa.

4. Scaffolding

Ruby on Rails secara otomatis membuat sebagian besar kode sementara yang diperlukan dalam tahap awal pengembangan aplikasi. Ini membantu pengembang dalam memulai pengembangan dengan cepat.

5. *Built-in Testing*

Ruby on Rails menyediakan tes otomatis sederhana yang dapat diperluas oleh pengembang. Ini termasuk dukungan kode yang memudahkan penulisan dan pelaksanaan kasus tes. Selain itu, tes otomatis dapat dijalankan dengan menggunakan utilitas *rake*.

6. Three environments development, testing, and production

Ruby on Rails menyediakan tiga komponen bawaan yaitu *development*, *testing*, dan *production*. Setiap komponen berbeda satu sama lain, tetapi dapat mempermudah seluruh siklus pengembangan perangkat lunak. Misalnya, *Ruby on Rails* membuat salinan *database test* yang baru untuk setiap uji coba.

7. Berbagai Fitur Tambahan

Ruby on Rails juga menyertakan berbagai fitur tambahan, termasuk *Ajax* untuk antarmuka pengguna yang kaya, tampilan parsial dan helper untuk menghemat penggunaan kode tampilan, caching bawaan, kerangka kerja pengiriman surat, dan layanan web.

Keunggulan-keunggulan ini membuat *Ruby on Rails* menjadi *framework* yang produktif untuk pengembangan aplikasi web, terutama dalam pengelolaan basis data dan penulisan kode yang efisien.

2.2.8 *PostgreSQL*

PostgreSQL adalah *database system* terbuka yang dapat diakses, dimodifikasi, dan digunakan tanpa pembatasan *propietary software* (*software* berlisensi), serta memberikan akses ke data *benchmarking* kinerja yang sering terbatas pada propietary *software* seperti *Oracle*, sehingga pengguna bebas menyesuaikannya sesuai kebutuhan [33]. Keunggulan dari *PostgreSQL* adalah sebagai berikut.

1. Object Relational DBMS

PostgreSQL menggunakan model data *object-relational* yang canggih, mampu mengatasi rutinitas dan peraturan kompleks. Ini termasuk dukungan untuk *declarative SQL query*, *multi-versi*on *concurrency control*, *multi-user-support*, transaksi, optimisasi kueri, *inheritance*, dan *arrays*.

2. Highly Extensible

PostgreSQL mendukung operator, fungsi, metode akses, dan tipe data yang dapat didefinisikan oleh pengguna. Ini memberikan fleksibilitas dalam mengadaptasi *database* sesuai kebutuhan.

3. Comprehensive SQL Support

PostgreSQL mendukung spesifikasi *SQL99* dan termasuk fitur-fitur lanjutan seperti kueri gabungan *SQL92*.

4. *Referential Integrity*

PostgreSQL mendukung integritas referensial untuk memastikan keabsahan data dalam *database*.

5. Flexible API

PostgreSQL memiliki antarmuka pemrograman aplikasi yang fleksibel, seperti O*bject Pascal, Python, Perl, ODBC, Java/JDBC, Ruby, TCL,* dan *Pike*, yang memungkinkan pengembang untuk mendukung *database PostgreSQL*.

6. *Procedural Languages*

PostgreSQL mendukung bahasa internal prosedural, termasuk bahasa asli yang disebut *PL*/*pgSQL*, yang mirip dengan bahasa prosedural *Oracle*, *PL*/*SQL*. Selain itu, PostgreSQL dapat menggunakan bahasa seperti *Perl*, *Python*, atau *TCL* sebagai bahasa prosedural yang tertanam.

7. Multi-Version Concurrency Control (MVCC)

PostgreSQL menggunakan teknologi *MVCC* untuk menghindari penahanan yang tidak perlu. Ini memungkinkan pembaca untuk tidak diblokir oleh penulis yang sedang memperbarui catatan dalam *database*.

8. Arsitektur *Client/Server*

PostgreSQL menggunakan arsitektur *process-per-user* yang mirip dengan *Apache* 1.3.x untuk mengelola koneksi pengguna ke *database*.

9. Write Ahead Logging (WAL)

Fitur *Write Ahead Logging (WAL)* meningkatkan keandalan *database dengan* mencatat perubahan sebelum ditulis ke *database*. Hal ini memastikan

bahwa dalam kasus kegagalan *database*, masih ada catatan transaksi yang dapat digunakan untuk pemulihan.

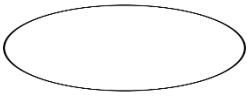
2.2.9 *Unified Modeling Language (UML)*

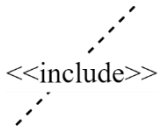
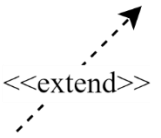
Unified Modeling Language (UML) merupakan sebuah bahasa yang berfungsi untuk mendefinisikan, memvisualisasikan, merancang, dan memberi dokumentasi komponen-komponen sistem perangkat lunak, seperti model, deskripsi, dan elemen perangkat lunak lainnya yang didasarkan pada konep orientasi objek untuk membantu dalam pemodelan sistem yang penggunaannya tidak terbatas pada perangkat lunak [34]. *UML* menggunakan berbagai jenis diagram untuk memvisualisasikan berbagai perspektif pada pengembangan sistem. Jenis-jenis diagram pada *UML* adalah sebagai berikut [34].

1. *Use Case* Diagram

Use Case Diagram adalah diagram untuk menggambarkan relasi antara pengguna dan sistem. *Use Case* Diagram bertujuan untuk memberikan gambaran tentang persyaratan fungsional dari sebuah produk termasuk dengan pengguna pada sistem atau yang disebut aktor. Berikut ini simbol-simbol pada *Use Case* Diagram dapat dilihat pada Tabel 2.2 [35].

Tabel 2.2 *Use Case Diagram*

Simbol	**Deskripsi**
	*Use Case* Menjelaskan fungsi dan kegunaan sistem yang dibuat.
	*Actor* Pengguna yang terlibat dan memiliki interaksi dengan sistem.
	*Association* Menghubungkan *use case* dengan *actor*

	*Include* Menunjukkan *use case* satu merupakan bagian dari *use case* lainnya.
	*Extend* Menunjukkan arah panah dengan putus-putus dari *use case* ke *base use case*.

2. *Class Diagram*

Class Diagram merupakan diagram yang menjelaskan relasi antar kelas pada suatu pengembangan sistem dan menunjukkan cara mereka bekerja sama. Diagram ini menunjukkan struktur dan mendeskripsikan *class*, package, dan objek yang saling berkaitan dengan *inheritance*, *associate*, dan lainnya. Berikut adalah simbol-simbol pada *Class* Diagram dapat dilihat pada Tabel 2.3.

Tabel 2.3 *Class Diagram*

Simbol	**Deskripsi**
Nama Kelas +Atribut +Metode	*Actor* Menggambarkan kelas pada sistem yang terbagi menjadi tiga bagian, yaitu nama kelas, atribut, dan metode.
	Association Hubungan statis antar kelas yang memberi gambaran atribut berupa kelas lain.
	*Agregation* Hubungan yang menyatakan suatu kelas menjadi atribut bagi kelas lain.
	Composition Bentuk khusus dari *agregation* di mana kelas yang menjadi bagian diciptakan setelah kelas *whole* dibuat.
	Generalization Relasi antar kelas dengan makna umum ke khusus.
	Directed Association Asosiasi yang memiliki makna kelas yang satu digunakan oleh kelas yang lain.

3. *Sequence Diagram*

Sequence Diagram digunakan untuk penggambaran interaksi antar objek yang terjadi pada sistem [36]. Diagram ini menunjukkan tahap yang dilakukan oleh aktor sampai dengan mendapatkan respons pada sistem. Berikut adalah simbol-simbol pada *Sequence Diagram* dapat dilihat pada Tabel 2.4.

Tabel 2.4 *Sequence Diagram*

Simbol	Deskripsi
	*Actor* Menggambarkan interaksi antara manusia dengan sistem
	*Object* Menggambarkan objek pada sistem
	Message Menggambarkan pengiriman suatu pesan
	*Replay Message* Menggambarkan pengiriman kembali suatu pesan
	Message Return Menggambarkan balasan dari pengiriman suatu pesan
	*A focus of cntrol and a life line* Menggambarkan mulai dan berakhirnya suatu pesan

2.2.10 Metode Pengembangan *Scrum*

Scrum adalah metodologi atau kerangka kerja terstruktur yang diterapkan untuk mengembangkan sistem yang memiliki kompleksitas tinggi dengan melakukan pendekatan bertahap untuk meningkatkan kemampuan prediksi dan mengendalikan risiko dalam pengembangan sistem [37]. Terdapat tiga peranan penting dalam metode *Scrum*, yaitu sebagai berikut [15].

1. *Product Owner*

Product Owner merupakan orang yang bertanggung jawab untuk berhubungan dengan *team development* terkait dengan tujuan dari proyek yang dirancang sehingga menghasilkan nilai bisnis dari produk yang dikembangkan.

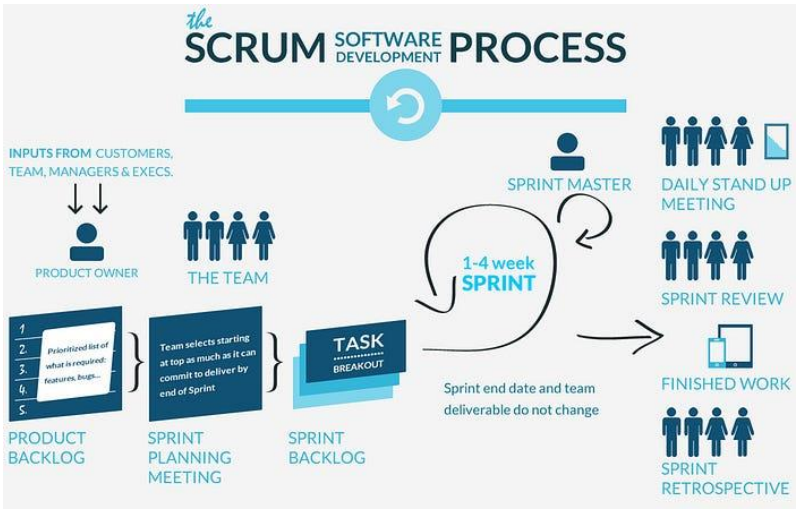
2. *Scrum Master*

Scrum Master berperan untuk menghubungkan antara *Product Owner* dan tim pengembang, *yang* terdiri atas *developer* dan *tester* (*Quality Assurance*). *Scrum Master* tidak memiliki kewenangan manajemen tim, tetapi membantu mengatasi masalah dan membantu tim mencapai target. *Scrum Master* dapat menyampaikan rekomendasi pada *Product Owner* untuk memaksimalkan *Return on Investment (ROI)* tim.

3. *Development Team*

Development Team bertanggung jawab atas aspek teknis pelaksanaan proyek. Terdapat beberapa tahapan pada pengembangan aplikasi menggunakan metode *Scrum*, antara lain [38]:

1. *product backlog,* bertujuan untuk menentukan prioritas yang akan dilakukan selama *sprint.*
2. *sprint planning,* tim berkumpul untuk mengidentifikasi tugas masing-masing sebagai perancanaan dalam mengerjakan *project.*
3. *sprint backlog,* masing-masing anggota tim membuat beberapa *task* sebagai daftar tugas yang akan dikerjakan selama satu periode *sprint.*
4. *daily scrum,* merupakan evaluasi harian terhadap tugas-tugas dan hambatan yang dihadapi tim selama *sprint*, berlangsung maksimal selama 15 menit.
5. *sprint review*, setiap anggota tim mempresentasikan hasil pekerjaan selama *sprint* berlangsung.
6. *sprint retrospective,* seluruh anggota tim memberikan pendapat serta evaluasi mengenai kinerja masing-masing anggota tim setelah *sprint* berlangsung pada metode *scrum*.



Gambar 2.1 Tahapan Metode *Scrum* untuk Pengembangan *Product* [39]

2.2.11 Metode *Black box Testing* (Pengujian kotak hitam)

Black box testing adalah metode pengujian fungsionalitas sistem tanpa melihat operasi internalnya [14]. Pengujian ini mempunyai tujuan untuk mengidentifikasi fungsi yang *error*, kesalahan pada *interface*, kesalahan struktur *database* eksternal, kinerja aplikasi yang melambat, dan kesalahan pada inisialisasi dan terminasi [40]. Pengujian *Black box* untuk memastikan bahwa fitur dan fungsi yang telah dibangun berjalan sesuai dengan yang diharapkan.

2.2.12 *Speed Test REST API*

Speed test adalah pengujian yang dilakukan untuk melihat performa dari *website* saat melakukan *load data* [41]. *Speed test* dilakukan dengan melakukan *test* terhadap data pada sistem dengan kondisi dan beban yang sama untuk masing-masing *endpoint* pada sistem [42]. Pengujian kecepatan akses pada *REST API* digunakan untuk melihat seberapa cepat pertukaran data yang terjadi pada *backend* ke *frontend*.