PERANCANGAN ARSITEKTUR APLIKASI BUDIDAYA PERIKANAN MODERN PADA BACKEND YANG BERTANGGUNG JAWAB DALAM MELAYANI TRANSAKSI QUERY WEBSERVICE DENGAN MENGGUNAKAN TEKNOLOGI FLASK MICROSERVICE

Proposal Skripsi

Disusun untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer

Oleh:
Andri Rahmanto R
1313618007

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA

2023

LEMBAR PENGESAHAN

Dengan ini saya mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta

Nama	:	Andri Rahmanto
No. Registrasi	:	1313618007
Program Studi	:	Ilmu Komputer
Judul	:	Perancangan Aritektur Aplikasi Budidaya Perikanan Modern Pada Backend yang Bertanggung Jawab Dalam Melayani Transaksi Query Webservice Dengan Menggunakan Teknologi Flask Micorservice

Menyatakan bahwa proposal skripsi ini telah siap diajukan untuk seminar pra skripsi.

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Eka Suryana, M.Kom.	**Med Irzal, M.Kom.**
NIP. 19851223 201212 1 002	NIP. 19770615 200312 1 001

Mengetahui,

Koordinator Program Studi Ilmu Komputer

Ir. Fariani Hermin Indiyah, M.T

NIP. 19600211 198703 2 001

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul **"Perancangan Aritektur Aplikasi Budidaya Perikanan Modern Pada Backend yang Bertanggung Jawab Dalam Melayani Transaksi Query Webservice Dengan Menggunakan Teknologi Flask Micorservice"**.

Keberhasilan dalam menyusun skripsi ini tidak lepas dari bantuan berbagai pihak yang mana dengan tulus dan ikhlas memberikan masukan guna sempurnanya skripsi ini. Oleh karena itu dalam kesempatan ini, dengan kerendahan hati penulis mengucapkan banyak terima kasih kepada:

1. Yth. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Ir. Fariani Hermin Indiyah, M.T selaku Koordinator Program Studi Ilmu Komputer.
3. Bapak Muhammad Eka Suryana M.Kom selaku Dosen Pembimbing I yang telah memberikan arahan dan bimbingan dalam penulisan proposal skripsi ini.
4. Bapak Med Irzal M.Kom selaku Dosen Pembimbing II yang telah memberikan arahan dan bimbingan dalam penulisan proposal skripsi ini.
5. Ayah dan Mama penulis yang selama ini telah mendukung dan membantu menyelesaikan skripsi ini.
6. Teman-teman Program Studi Ilmu Komputer 2018 yang telah mendukung dan membantu skripsi ini.

Penulis menyadari bahwa penyusunan skripsi ini masih jauh dari sempurna karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karenanya, kritik dan

saran yang bersifat membangun akan penulis terima dengan senang hati. Akhir kata, penulis berharap tugas akhir ini bermanfaat bagi semua pihak khususnya penulis sendiri. Semoga Allah SWT senantiasa membalas kebaikan semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini.

Jakarta, 1 Febuari 2023

Andri Rahmanto

DAFTAR ISI

DAFTAR ISI vi

DAFTAR GAMBAR vii

DAFTAR TABEL viii

I PENDAHULUAN 1

A. Latar Belakang Masalah 1

B. Rumusan Masalah 4

C. Pembatasan Masalah 5

D. Tujuan Penelitian 5

E. Manfaat Penelitian 5

II KAJIAN PUSTAKA 7

A. *Scrum* 7

1. Tim Scrum 8

2. Aktivitas-aktivitas Scrum (Scrum Events) 11

3. Komponen *Scrum* (*Scrum Artifacts*) 15

B. REST 17

1. URL Design 18

2. HTTP Verbs 18

3. HTTP Response Code 19

4. Format Response 19

C. MongoDB 19

D. Flask Framework 23

1. Flask routing 23

2. Dependencies 24
E. Apache 27
1. Konfigurasi httpd.conf 27

III Metodologi Penelitian **29**
A. Pengumpulan Data 30
B. Analisa Kebutuhan 30
C. Analisa Sistem 31
D. Perancangan Sistem 33
1. Produk Backlog 34
2. Sprint Backlog 36
3. Daily Scrum 37
4. Sprint Review dan Sprint Retrospective 37
5. Sprint 1 Report 38
6. Meeting Sprint 2 47

LAMPIRAN **48**

A Transkrip Percakapan **48**

DAFTAR PUSTAKA **51**

DAFTAR GAMBAR

Gambar 2.1 Pemodelan RDBMS 20

Gambar 2.2 Pemodelan NoSQL 20

Gambar 3.1 Tahapan Peneltian 29

Gambar 3.2 *Flowchart* request 31

Gambar 3.3 *Usecase* diagram 32

Gambar 3.4 Flowchart Scrum 33

Gambar 3.5 Github Projects Sprint-1 39

Gambar 3.6 ERD Database Sprint-1 40

Gambar 3.7 Class Diagram Sprint-1 42

Gambar 3.8 *Request Get* pemberian pakan *Postman* 46

DAFTAR TABEL

Tabel 2.1 *Komparasi Database RDBMS dan NoSQL* 22

Tabel 3.1 *Product Backlog* 35

Tabel 3.2 *Sprint-1 backlog* 37

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Indonesia memiliki beberapa sumber perikanan salah satunya adalah budidaya ikan air tawar. Budidaya ikan air tawar memerlukan beberapa modal yaitu lahan, prasarana kolam, Pakan, dan Pengetahuan atau skill dalam berbudidaya ikan. Umumnya budidaya ikan air tawar memelihara banyak jenis ikan atau yang disebut pemeliharaan campuran hal ini disebabkan karena terdapatnya berbagai macam makanan untuk berbagai jenis ikan air tawar, walaupun lebih baik memperhatikan mana jenis ikan yang akan dijadikan peliharaan pokok dan mana yang akan dijadikan peliharaan sampingan (infishta, 2019).

Dalam pembudidayaan ikan air tawar diperlukannya pencatatan beberapa indikator yang berguna untuk menilai kelayakan habitat, memantau perkembangan ikan, dan memantau pemberian pakan sehingga bisa memutuskan treatment apa yang akan dilakukan kedepannya untuk mencapai panen yang baik. Beberapa indikator yang perlu di perhatikan adalah kadar oksigen dalam air, pengukuran kadar ph dalam air, DO (Dissolved Oxygen), Suhu, dan Ammonia. Indikator air tersebut berpengaruh terhadap kelayakan habitat ikan air tawar (Pramleonita dkk., 2018). Di sisi lain pencatatan terhadap kolam, perlakuan kolam, penaburan ikan dan pemberian pakan juga merupakan faktor yang penting untuk dicatat. Perlunya pencatatan terhadap banyak indikator secara berulang dan intensif menjadi faktor diperlukannya sistem modern yang membantu mempermudah pekerjaan tersebut.

Di Indonesia terdapat beberapa penelitian sistem modern terkait perikanan yang berkontribusi pada budidaya perikanan oleh (Supriyati, 2018) melakukan

penelitian yang menghasilkan sistem akuntansi budidaya perikanan berbasis android. Penelitian ini dilakukan untuk memecahkan suatu masalah pada pencatatan secara tradisional yang dilakukan oleh petani perikanan yang menyebabkan kurang maksimalnya income yang didapat. Sistem yang dibuat akan digunakan oleh beberapa stakeholders diantaranya adalah: Ketua dari perikanan tersebut, Sekertaris, Bendahara, Seksi sarana produksi, Seksi pengadaan, dan Seksi pemasaran. Sistem ini menerapkan beberapa fitur yaitu Transaksi jual-beli, kas keluar-masuk, dan pembukuan. (Dhita Widhiastika, 2021) melakukan penelitian yang berjudul "Perancangan Aplikasi Jual Beli Produk Perikanan Berbasis Mobile Android". Aplikasi tersebut memungkinkan pengguna dalam melakukan transaksi jual beli perikanan. Aplikasi juga menyediakan wadah untuk diskusi dan informasi gizi dari suatu produk perikanan. Dari dua penelitian sebelumnya didapatkan sistem yang dibuat membantu petani perikanan dalam kegiatan pembukuan transaksi, namun tidak dengan kegiatan aktivitas budidaya.

Penelitian pada sektor IoT pemantauan kualitas air secara real-time dilakukan oleh (Yudhis Thiro Kabul Yunior, 2019) yang mana menghasilkan IoT monitoring kualitas air terintegrasi oleh database server. Indikator yang dicatat adalah Ph, Dissolved Oxygen (DO), Suhu, dan Turbidity. IoT dibuat menggunakan microcontroller Arduino, yang mana perannya adalah mengirim data dari sensor dengan protokol GSM ke database server. Pada server juga terdapat halaman yang menampilkan monitoring data berdasarkan database. Penelitian (Ahmad Rifa'i, 2021) menghasilkan IoT untuk Pemantauan dan Kontrol otomatis kualitas air berbasis IoT menggunakan Node-Red untuk budidaya udang. Dalam penerapannya indikator yang diukur sama dengan apa yang diukur dalam penelitian (Yudhis Thiro Kabul Yunior, 2019) namun dengan tambahan pengukuran ketinggian air. Nantinya setiap sensor terhubung dengan microcontroller arduino, namun pada penelitian

(Ahmad Rifa'i, 2021) memilih menggunakan NodeMCU (modul wifi) untuk terhubung ke internet. Server yang dibangun menggunakan Node-Red yang merupakan tools yang mempermudah membangun sistem berbasis IoT. Saat pengiriman data ke server arduino menggunakan protokol MQTT yang menerapkan konsep publish dan subscribe. Pada alat kontrol aktuator terdiri dari NodeMCU dan relay dan berguna sebagai pengontrol pompa air dan aerator. Dari dua penelitian diatas terdapat beberapa masalah terkait device IoT terutama pada sensor Ph dan DO. Keakuratan sensor Ph dan DO didasarkan pada grade sensor, semakin tinggi grade sensor semakin mahal sensor tersebut. Sensor Ph dan Do juga mengalami penurunan kualitas seiring waktu berjalan, maka dari itu diperlukannya replacement berkala selambat-lambatnya 1 bulan sekali.

Penelitian IoT pemberian pakan otomatis dilakukan oleh (Aep Setiawan, 2022) yang menghasilkan suatu alat pemberian pakan otomatis. Alat pemberian pakan dapat bekerja secara otomatis berdasarkan waktu yang telah di jadwalkan. IoT yang diterapkan menggunakan NodeMCU sebagai modul yang bisa mengakses jaringan wifi dan blynk sebagai middleware untuk memonitoring pemberian pakan, persediaan pakan,pengaturan jadwal dan takaran pakan. Dalam kasus IoT pemberian pakan otomatis terdapat beberapa kendala yaitu resiko tidak meratanya pemberian pakan karena terbatasnya jangkauan pelemparan pakan. Selanjutnya adalah pelemparan pakan otomatis pada praktiknya ditetapkan oleh jadwal sedangkan dosis pakan bisa berubah tergantung dengan keadaan di lapangan.

Dari beberapa penelitian diatas, penelitian ini bertujuan untuk memberikan solusi dari setiap permasalahan yang ada pada perikanan modern. Fokus penelitian ini adalah membangun arsitektur aplikasi budidaya perikanan modern pada backend yang dapat melayani transaksi query dari berbagai platform. Penelitian ini selanjutnya akan terus dikembangkan dari berbagai sisi antara lain adalah frontend,

AI, dll. Berikut beberapa penelitian yang telah dilakukan untuk berkontribusi pada perikanan modern ini.

Penelitian yang terkait dalam penelitian ini adalah penelitian bidang Monitoring data sensing pada budidaya ikan air tawar sudah dilakukan oleh Fadhilah Perwira Hadi dalam penelitian yang berjudul "Rancang Bangun Web Service dan Website sebagai Storage Engine dan Monitoring Data Sensing". Penelitian tersebut menghasilkan suatu sistem web service yang dapat menerima dan memonitoring data yang dikirimkan oleh embedded device, dengan menerapkan konsep IoT (Hadi, 2021). Pada dasarnya web service yang dirancang oleh (Hadi, 2021) dikhususkan untuk tersambung kepada embedded device IoT. Embedded device sebagai perangkat sensor terhadap beberapa data-data indikator unsur dalam kolam serta mengirim data tersebut ke web service. Penelitian lainnya yang terkait adalah penelitian yang dilakukan oleh (Nugraha, 2022) yang berjudul "Ekstraksi Latar Depan pada Citra Ikan dengan Metode GrabCut yang Diautomasi Menggunakan Saliency Map" yang bertujuan untuk membangun sistem yang memisahkan antara latar depan dan latar belakang pada citra ikan. Penelitian selanjutnya adalah penelitian dengan judul "Fish Movement Tracking dengan Menggunakan Metode GMM dan Kalman Filter" yang dilakukan oleh (Alim, 2022) yang bertujuan untuk membangun sebuah sistem yang dapat melakukan pelacakan pergerakan ikan yang diharapkan nantinya dapat dikembangkan kembali untuk sistem penghitungan ikan. Ke 3 penelitian tersebut merupakan kontribusi yang kedepannya akan diterapkan bersamaan dalam penelitian ini kedalam sistem induk.

B. Rumusan Masalah

Berdasarkan latar belakang penelitian ini, maka perumusan masalah pada penelitian ini adalah “Bagaimana merancang Arsitektur Aplikasi Budidaya

Perikanan Modern pada Backend yang Bertanggung Jawab dalam Melayani Transaksi Query Webservice dengan Menggunakan Teknologi Flask Microservice".

C. Pembatasan Masalah

Adapun beberapa pembatasan masalah yang bertujuan agar penelitian ini lebih terarah dan sesuai dengan tujuan penelitian:

1. Webservice dikembangkan khusus untuk 1 mitra, dalam hal ini UD JFarm Teknologi. Hanya untuk 1 user.
2. Pengembangan web service menggunakan Framework Flask.

D. Tujuan Penelitian

Penelitian yang dilakukan bertujuan untuk merancang arsitektur aplikasi budidaya perikanan modern pada backend yang bertanggung jawab dalam melayani transaksi query webservice.

E. Manfaat Penelitian

1. Bagi sektor perikanan

   Hasil perancangan arsitektur backend server ini dapat memberikan kontribusi terhadap sistem perikanan modern dalam bentuk web service yang dapat terhubung dengan multi platform.

2. Bagi penulis

   Menambah pengetahuan dibidang pengembangan web service khususnya pengembangan Arsitektur BackEnd REST API, mengasah kemampuan *programming*, dan memperoleh gelar sarjana dibidang Ilmu Komputer. Selain

itu, penulisan ini juga merupakan media bagi penulis untuk mengaplikasikan ilmu yang didapat di kampus ke kehidupan masyarakat.

3. Bagi Universitas Negeri Jakarta

   Menjadi pertimbangan dan evaluasi akademik khususnya Program Studi Ilmu Komputer dalam penyusunan skripsi sehingga dapat meningkatkan kualitas akademik di program studi Ilmu Komputer Universitas Negeri Jakarta serta meningkatkan kualitas lulusannya.

BAB II

KAJIAN PUSTAKA

A. *Scrum*

Scrum merupakan kerangka kerja ringan yang dapat menghasilkan solusi yang adaptif untuk masalah yang kompleks bagi orang, tim, dan organisasi. Kerangka kerja scrum dibiarkan tidak lengkap, hanya mendefinisikan bagian-bagian yang diperlukan untuk mengimplementasi teori *scrum*, selebihnya kecerdasan kolektif orang-orang yang menggunakan *scrum* akan membangun *scrum* itu sendiri. *Scrum* tidak memberikan instruksi yang terperinci kepada setiap anggota tim, melainkan memandu hubungan dan interaksi mereka.

Scrum terus berkembang dengan adanya pengalaman dan pemikiran, lalu menghasilkan suatu keputusan berdasarkan pengamatan. Menggunakan proses iterasi dan perlahan menambahkan sesuatu, untuk mengendalikan risiko dan membuat sesuatu dapat diprediksi.

Scrum menggunakan prinsip pendekatan *Agile* untuk dapat mengatasi segala macam masalah secara kreatif dan adaptif. Berbagai proses, teknik dan metode dapat digunakan dalam kerangka kerja.

Scrum menggabungkan empat aktivitas formal untuk memeriksa dan mengadaptasi menjadi satu aktivitas konten, *Sprint*. Kegiatan ini dapat berhasil jika menerapkan pilar empiris transparansi, inspeksi, dan adaptasi *Scrum*.

1. Transparansi

   Proses dan pekerjaan yang muncul harus dapat dilihat oleh mereka yang melakukan pekerjaan dan mereka yang menerimanya. Untuk *Scrum*, sangat penting bahwa keputusan didasarkan pada kondisi tiga komponen formal, dan

komponen *Scrum* dengan transparansi rendah mengurangi nilai dan meningkatkan risiko.

2. Inspeksi

   Artefak *scrum* dan kemajuan menuju tujuan yang disepakati harus dirombak secara berkala untuk mendeteksi kemungkinan perbedaan atau masalah yang tidak terduga. Untuk membantu inspeksi, *Scrum* menyediakan ritme dalam bentuk lima acaranya.

   Centang untuk memungkinkan adaptasi. Inspeksi tanpa adaptasi dianggap tidak berguna. Aktivitas *scrum* dirancang untuk menginspirasi perubahan.

3. Adaptasi

   Jika ada aspek proses yang menyimpang dari batas yang dapat diterima, atau jika produk yang dihasilkan tidak dapat diterima, proses yang diterapkan atau bahan yang dihasilkan harus disesuaikan. Penyesuaian harus dilakukan sesegera mungkin untuk meminimalkan penyimpangan lebih lanjut.

   Adaptasi menjadi lebih sulit ketika orang-orang yang terlibat tidak diberdayakan atau dikelola sendiri. Tim *scrum* diharapkan untuk menyesuaikan diri saat mereka mempelajari hal-hal baru melalui inspeksi.

1. Tim Scrum

Unit dasar *Scrum* adalah tim kecil. *Scrum Team* terdiri dari *Scrum Master*, *Product Owner*, dan *Developer*. Dalam tim *Scrum*, tidak ada sub-tim atau hierarki. Tim *scrum* bersifat lintas fungsi, yang berarti bahwa anggota memiliki semua keterampilan yang mereka butuhkan untuk menciptakan nilai di setiap *Sprint*. Tim scrum cukup kecil untuk tetap lincah dan cukup besar untuk menyelesaikan pekerjaan penting dalam sprint, biasanya 10 orang atau kurang.

Tim *Scrum* bertanggung jawab atas semua aktivitas terkait produk, termasuk kolaborasi pemangku kepentingan, validasi, pemeliharaan, operasi, eksperimen, penelitian dan pengembangan, dan aktivitas lain apa pun yang mungkin diperlukan.

Seluruh Tim *Scrum* bertanggung jawab untuk menciptakan peningkatan yang berharga dan berguna di setiap *Sprint*. *Scrum* mendefinisikan tiga tanggung jawab khusus dalam Tim *Scrum*: Pengembang, Pemilik Produk, dan *Scrum Master*.

1. Pengembang (*Developers*)

   Pengembang adalah seseorang di tim *Scrum* yang bekerja untuk membuat aspek apa pun dari *Increment* yang dapat digunakan oleh setiap Sprint.

   Keterampilan khusus yang dibutuhkan oleh pengembang seringkali luas dan bervariasi menurut bidang pekerjaan. Namun, pengembang selalu bertanggung jawab untuk:

   a. Mengembangkan rencana untuk *Sprint*, *Sprint Backlog*;
   b. Menanamkan kualitas dengan tetap berpegang pada definisi selesai;
   c. Menyesuaikan rencana harian mereka untuk mencapai *Sprint Goals*; dan,
   d. Sebagai profesional, kami memikul tanggung jawab satu sama lain.

2. Pemilik Produk (*Product Owner*)

   Pemilik Produk bertanggung jawab untuk memaksimalkan nilai produk yang dikembangkan oleh Tim *Scrum*. Bagaimana hal ini dicapai akan bervariasi antara perusahaan, tim, dan individu. *Product Owner* juga bertanggung jawab untuk mengelola *Product Backlog* dengan baik, termasuk:

   a. Mengembangkan dan mengkomunikasikan tujuan produk dengan jelas;
   b. Membuat dan mengkomunikasikan item backlog produk dengan jelas;

c. Memesan backlog produk; dan,

d. Pastikan *Product Backlog* transparan, terlihat, dan mudah dipahami.

Pemilik Produk dapat melakukan pekerjaan yang dijelaskan di atas atau mendelegasikan tanggung jawab kepada orang lain. Seluruh organisasi harus menghormati keputusan mereka. Keputusan ini tercermin dalam konten dan urutan *Product Backlog*, dan melalui peningkatan yang dapat diperiksa di *Sprint Review*.

Pemilik Produk adalah orang, bukan panitia. Pemilik Produk dapat mewakili kebutuhan banyak pemangku kepentingan dalam *Product Backlog*. Siapapun anggota tim yang ingin mengubah *Product Backlog* dapat melakukannya dengan mencoba meyakinkan Pemilik Produk.

3. *Scrum Master*

Scrum Master bertanggung jawab untuk membangun *Scrum* sebagaimana didefinisikan dalam Panduan *Scrum*. Mereka melakukan ini dengan membantu semua orang di tim dan organisasi *Scrum* memahami teori dan praktik *Scrum*.

Scrum Master bertanggung jawab atas efektivitas Tim *Scrum*. Mereka melakukan ini dengan meminta tim *Scrum* meningkatkan praktik mereka dalam kerangka kerja *Scrum*.

Scrum Masters adalah pemimpin sejati yang melayani Tim *Scrum* dan organisasi yang lebih besar.

Scrum Master melayani Tim Scrum dengan cara, seperti:

a. Melatih anggota tim dalam manajemen diri dan manajemen berbagai bidang;

b. Bantu tim *Scrum* fokus pada penciptaan peningkatan bernilai tinggi yang memenuhi definisi selesai;

c. Buka blokir kemajuan Tim *Scrum*; dan,

d. Pastikan semua acara *Scrum* terjadi, positif, produktif, dan sesuai jadwal.

Scrum Master melayani Pemilik Produk dengan cara, seperti:

a. Membantu menemukan teknik untuk definisi tujuan produk yang efektif dan manajemen simpanan produk;

b. Membantu tim *Scrum* memahami kebutuhan akan item *Product Backlog* yang jelas dan ringkas;

c. Membantu membangun program produk empiris untuk lingkungan yang kompleks; dan,

d. Memfasilitasi kolaborasi pemangku kepentingan sesuai kebutuhan atau kebutuhan.

Scrum Master melayani organisasi dengan cara, seperti:

a. Memimpin, melatih dan melatih organisasi dalam implementasi *Scrum*;

b. Merencanakan dan merekomendasikan penerapan *Scrum* dalam organisasi;

c. Membantu karyawan dan pemangku kepentingan memahami dan menerapkan metode empiris untuk pekerjaan yang kompleks; dan,

d. Menghilangkan hambatan antara *stakeholder* dan tim *Scrum*.

2. Aktivitas-aktivitas Scrum (Scrum Events)

Sprint adalah wadah untuk semua acara lainnya. Setiap acara di *Scrum* adalah kesempatan formal untuk meninjau dan menyesuaikan artefak *Scrum*.

Acara-acara ini secara khusus dirancang untuk mencapai transparansi yang diperlukan. Kegagalan untuk mengoperasikan acara apapun seperti yang ditentukan menghasilkan kesempatan yang hilang untuk peninjauan dan adaptasi. Acara digunakan di *Scrum* untuk menciptakan ketertiban dan meminimalkan kebutuhan akan rapat yang tidak ditentukan dalam *Scrum*. Idealnya, semua acara diadakan pada waktu dan tempat yang sama untuk mengurangi kerumitan.

1. *Sprint*

   Sprint adalah jantung dari *Scrum*, di mana ide-ide diubah menjadi nilai. Mereka adalah acara berdurasi tetap satu bulan atau kurang untuk menciptakan konsistensi. *Sprint* baru dimulai segera setelah *Sprint* sebelumnya berakhir.

   Semua pekerjaan yang diperlukan untuk mencapai Tujuan Produk, termasuk Perencanaan *Sprint*, *Scrum* Harian, Ulasan *Sprint*, dan Retrospektif *Sprint*, terjadi selama:

   a. tidak membuat perubahan apa pun yang akan membahayakan *Sprint Goal*;
   b. Kualitas tidak berkurang;
   c. *Product Backlog* sedetail yang dibutuhkan; dan,
   d. Semakin banyak yang dipelajari, ruang lingkup dapat diklarifikasi dan dinegosiasikan ulang dengan Pemilik Produk.

   Sprint memungkinkan prediktabilitas dengan memastikan bahwa kemajuan terhadap sasaran produk diperiksa dan diakomodasi setidaknya setiap bulan kalender. Ketika Cakrawala Sprint terlalu panjang, Tujuan Sprint mungkin gagal, kompleksitas dapat meningkat, dan risiko dapat meningkat. *Sprint* yang lebih pendek dapat digunakan untuk menghasilkan lebih banyak siklus

pembelajaran dan membatasi risiko biaya dan upaya untuk kerangka waktu yang lebih pendek. Setiap *Sprint* dapat dianggap sebagai proyek pendek.

Berbagai praktik ada untuk memprediksi kemajuan, seperti kelelahan, kelelahan, atau aliran kumulatif. Meskipun terbukti bermanfaat, ini tidak menggantikan pentingnya empirisme. Dalam lingkungan yang kompleks, apa yang terjadi tidak diketahui. Hanya apa yang telah terjadi yang dapat digunakan untuk keputusan berwawasan ke depan. *Sprint* dapat dibatalkan jika *Sprint Goals* sudah kedaluwarsa. Hanya Pemilik Produk yang berhak membatalkan Sprint.

2. Perencanaan *Sprint*

   Perencanaan *Sprint* memulai *Sprint* dengan meletakkan pekerjaan yang harus dilakukan untuk *Sprint*. Rencana akhir dibuat secara kolaboratif oleh seluruh tim *Scrum*.

   Product Owner memastikan bahwa peserta siap untuk mendiskusikan item *Product Backlog* yang paling penting dan bagaimana mereka memetakan ke tujuan produk. Tim *Scrum* juga dapat mengundang orang lain untuk berpartisipasi dalam Perencanaan *Sprint* untuk memberikan saran.

3. *Scrum* harian (*Daily Scrum*)

   Tujuan dari *Daily Scrum* adalah untuk memeriksa kemajuan terhadap *Sprint Goals* dan menyesuaikan *Sprint Backlog* sesuai kebutuhan untuk menyesuaikan rencana kerja di masa mendatang.

   Daily Scrum adalah aktivitas 15 menit untuk pengembang di tim *Scrum*. Untuk mengurangi kompleksitas, ini dilakukan pada waktu dan tempat yang sama setiap hari kerja *Sprint*. Jika *Product Owner* atau *Scrum Master* aktif mengerjakan item di *Sprint Backlog*, mereka berpartisipasi sebagai *developer*.

Pengembang dapat memilih struktur dan teknik apa pun yang mereka inginkan, selama *Scrum* harian mereka berfokus pada kemajuan Tujuan *Sprint* dan memiliki rencana yang dapat diterapkan untuk hari kerja berikutnya. Ini menciptakan fokus dan meningkatkan manajemen diri.

Daily Scrum dapat meningkatkan komunikasi, mengidentifikasi hambatan, dan memfasilitasi pengambilan keputusan yang cepat, menghilangkan kebutuhan untuk rapat ulang.

Daily Scrum bukan satu-satunya waktu pengembang menyesuaikan rencana mereka. Tim sering bertemu sepanjang hari untuk berdiskusi lebih rinci tentang cara menyesuaikan atau merencanakan ulang sisa *print*.

4. Pembahasan *Sprint* (*Sprint Review*)

   Tujuan dari *Sprint Review* adalah untuk memeriksa hasil *Sprint* dan mengidentifikasi penyesuaian di masa depan. Tim scrum mempresentasikan hasil kerja mereka kepada pemangku kepentingan utama dan mendiskusikan kemajuan menuju tujuan produk.

   Selama acara, Tim *Scrum* dan pemangku kepentingan meninjau apa yang telah dicapai dalam *Sprint* dan bagaimana lingkungan mereka telah berubah. Berdasarkan informasi ini, peserta berkolaborasi tentang apa yang harus dilakukan selanjutnya. *Product Backlog* juga dapat disesuaikan untuk memenuhi peluang baru. Tinjauan *Sprint* adalah rapat kerja dan Tim Scrum harus menghindari membatasinya pada presentasi.

   Sprint Review adalah kegiatan *Sprint* kedua dari belakang, dan *Sprint* sebulan tidak melebihi maksimal empat jam. Acara biasanya lebih pendek untuk *Sprint* yang lebih pendek.

5. *Sprint Retrospektif*

Tujuan dari *Sprint Retrospective* adalah untuk merencanakan cara-cara untuk meningkatkan kualitas dan efektivitas.

Tim *Scrum* memeriksa bagaimana *Sprint* terakhir berjalan sehubungan dengan individu, interaksi, proses, alat, dan Definisi Selesai. Elemen yang diperiksa seringkali berbeda dengan domain pekerjaan. Asumsi yang menyesatkan mereka diidentifikasi dan asal-usulnya dieksplorasi. Tim *Scrum* mendiskusikan apa yang berjalan dengan baik selama *Sprint*, masalah apa yang dihadapi, dan bagaimana masalah tersebut (atau tidak) diselesaikan.

Tim *Scrum* mengidentifikasi perubahan yang paling membantu untuk meningkatkan efektivitasnya. Perbaikan yang paling berdampak ditangani sesegera mungkin. Mereka bahkan dapat ditambahkan ke *Sprint Backlog* untuk *Sprint* berikutnya.

Sprint Retrospective mengakhiri *Sprint*. Ini dibatasi waktu hingga maksimum tiga jam untuk *Sprint* satu bulan. Untuk *Sprint* yang lebih pendek, acaranya biasanya lebih pendek.

3. Komponen *Scrum* (*Scrum Artifacts*)

Artefak *scrum* mewakili pekerjaan atau nilai. Mereka dirancang untuk memaksimalkan transparansi informasi penting. Jadi setiap orang yang mengkajinya memiliki dasar yang sama untuk adaptasi.

Setiap artefak berisi komitmen untuk memastikan bahwa artefak tersebut memberikan informasi yang meningkatkan transparansi dan fokus serta kemajuan yang dapat diukur:

a. Untuk *Product Backlog*, itu adalah tujuan produk.

b. Untuk *Sprint Backlog*, itu adalah *Sprint Goal*.

c. Untuk *Increment*, itu adalah definisi penyelesaian.

Komitmen ini dirancang untuk memperkuat empirisme dan nilai-nilai *Scrum* bagi tim *Scrum* dan pemangku kepentingannya.

a. *Product Backlog*

Product Backlog adalah daftar berurutan yang muncul dari apa yang dibutuhkan untuk meningkatkan suatu produk. Ini adalah satu-satunya sumber pekerjaan yang dilakukan oleh tim *Scrum*.

Item *Product Backlog* yang dapat dikerjakan oleh *Scrum Team* dalam sebuah *Sprint* dianggap siap untuk diseleksi dalam aktivitas Sprint Planning. Mereka biasanya mendapatkan transparansi ini setelah kegiatan pemurnian. Penyempurnaan *Product Backlog* adalah tindakan memecah dan mendefinisikan lebih lanjut item *Product Backlog* menjadi item yang lebih kecil dan lebih tepat. Menambahkan detail seperti deskripsi, urutan, dan dimensi adalah aktivitas yang berkelanjutan. Atribut sering bervariasi menurut bidang pekerjaan.

Pengembang yang akan melakukan pekerjaan bertanggung jawab atas ukurannya. Pemilik produk dapat mempengaruhi pengembang dengan membantu mereka memahami dan memilih pertukaran.

b. *Sprint Backlog*

Sprint Backlog mencakup *Sprint Goal* (Mengapa), satu set Item *Product Backlog* yang dipilih untuk *Sprint* (Apa), dan rencana yang layak untuk memberikan *Increment* (Bagaimana).

Sprint Backlog adalah rencana yang dibuat oleh pengembang. Ini adalah gambaran *real-time* yang sangat visual tentang apa yang akan dicapai pengembang selama *Sprint* untuk mencapai Tujuan *Sprint*. Oleh karena itu, *Sprint Backlog* diperbarui

sepanjang *Sprint* karena lebih banyak yang dipelajari. Itu harus memiliki detail yang cukup sehingga mereka dapat memeriksa kemajuan mereka di *Scrum* harian.

c. *Increment* (Peningkatan)

Increment adalah batu loncatan konkret untuk mencapai tujuan produk. Setiap kenaikan melengkapi semua kenaikan sebelumnya dan divalidasi secara menyeluruh untuk memastikan bahwa semua kenaikan bekerja sama. Untuk memberikan nilai, peningkatan harus tersedia.

Mungkin ada beberapa peningkatan dalam *Sprint*. Jumlah kenaikan yang disediakan dalam *Sprint Review* mendukung empirisme. Namun, delta dapat dikirimkan ke pemangku kepentingan sebelum akhir *sprint*. Ulasan *sprint* tidak boleh dilihat sebagai cara untuk membuka nilai.

Pekerjaan tidak dapat dianggap sebagai bagian dari kenaikan kecuali jika memenuhi definisi selesai.

B. REST

REST adalah arsitektur yang digunakan untuk pemrograman web, dan beroperasi menggunakan protokol HTTP. Arsitektur REST berfokus pada sumber daya, yang masing-masing merupakan komponen terpisah. Sumber daya diakses melalui tautan menggunakan metode HTTP standar, dan setiap klien dan server dalam sistem REST bekerja secara berbeda, klien mengakses dan berinteraksi dengan sumber daya, dan server hanya menyediakan akses ke sumber daya.

Arsitektur REST adalah arsitektur pengembangan API yang menggunakan hyperlink, header, dan kode status untuk berkomunikasi. Layanan web RESTful adalah layanan web yang dikembangkan menggunakan arsitektur REST. Setiap sumber daya di REST diberi URI atau ID universal. Layanan web RESTful

menggunakan metode HTTP untuk menyediakan representasi sumber daya, seperti yang didefinisikan oleh Uniform Resource Identifier. Layanan web menggunakan metode HTTP untuk mengimplementasikan arsitektur REST.

1. URL Design

Akses RESTful API menggunakan protokol HTTP. Penamaan dan struktur URL yang konsisten akan menghasilkan API yang baik yang mudah dipahami oleh pengembang. URL API sering disebut sebagai titik akhir dalam pemanggilannya.

2. HTTP Verbs

Setiap permintaan yang dibuat di sana menggunakan metode sehingga server tahu apa yang diminta klien:

a. GET

GET adalah metode permintaan HTTP paling sederhana yang digunakan untuk membaca atau mendapatkan data dari suatu sumber.

b. POST

POST adalah metode permintaan HTTP yang digunakan untuk membuat data baru dengan memasukkan data ke dalam badan saat membuat permintaan.

c. PUT

PUT adalah metode permintaan HTTP yang biasanya digunakan untuk memperbarui data sumber daya.

d. DELETE

DELETE adalah metode permintaan HTTP yang digunakan untuk menghapus data dari sumber daya.

3. HTTP Response Code

Kode respons HTTP adalah kode standar yang memberitahu klien tentang hasil permintaan. Secara umum, ada 3 kode grup yang sering dijumpai di RESTful API, yaitu:

a. 2XX : adalah kode respon yang menunjukkan bahwa permintaan berhasil.

b. 4XX : adalah kode respon yang menunjukkan bahwa permintaan mengalami kesalahan di sisi klien.

c. 5XX: adalah kode respon, yang menunjukkan bahwa permintaan mengalami kesalahan di sisi server.

4. Format Response

Setiap permintaan yang dibuat, klien akan menerima data respons dari server, biasanya dalam bentuk data XML atau JSON. Setelah klien memiliki data respons, klien dapat menggunakannya dengan menguraikan data dan memprosesnya sesuai kebutuhan.

C. MongoDB

MongoDB adalah sistem manajemen basis data yang dirancang untuk internet dan aplikasi berbasis web. MongoDB adalah database NoSQL berbasis dokumen. Model data dokumen MongoDB membuatnya mudah untuk dibuat aktif, karena memiliki dukungan untuk data tidak terstruktur. Dokumen MongoDB dikodekan dalam format seperti JSON, yang disebut BSON. BSON sangat cocok untuk objek modern metodologi pemrograman berorientasi dan ringan dan cepat (Hema Krishnan, 2016).

Perbedaan utama antara NoSQL dan *Relational Database Management System* (RDBMS) adalah bagaimana data dimodelkan pada basis data. Pemodelan pada RDBMS masih menggunakan tabel-tabel terstruktur, dimana kolom-kolom pada setiap barisnya tetap satu sama lain, sedangkan pemodelan data pada NoSQL khususnya pada MongoDB menggunakan dokumen-dokumen dimana setiap baris memiliki Kolom-kolom yang berbeda dengan baris lainnya. Contoh perbedaan konseptual database antara RDBMS dan NoSQL ditunjukkan pada **Gambar 2.1** (Pemodelan RDBMS) dan **Gambar 2.2** (Pemodelan NoSQL).

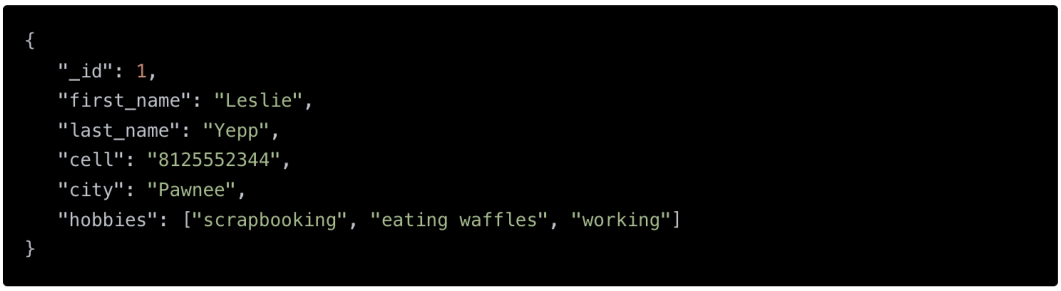
Users

ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee

Hobbies

ID	user_id	hobby
10	1	scrapbooking
11	1	eating waffles
12	1	working

Gambar 2.1: Pemodelan RDBMS
Sumber: *https://www.mongodb.com/nosql-explained*



Gambar 2.2: Pemodelan NoSQL
Sumber: *https://www.mongodb.com/nosql-explained*

Menurut Gambar 2.1, untuk menyimpan data *user* dan data *hobby*, diperlukan dua tabel terpisah, sedangkan dalam pemodelan NoSQL Gambar 2.2, data *user* dan *hobby* dalam satu dokumen yang sama. Dengan NoSQL, ketika ingin mengambil dua bagian data secara bersamaan, hanya memerlukan satu dokumen tanpa gabungan, yang menghasilkan kueri lebih cepat daripada RDBMS.

Meskipun NoSQL, desain database masih harus dilakukan. Desain basis data memungkinkan untuk mengonfirmasi dan mendokumentasikan pemahaman tentang basis data dan memastikan bahwa orang-orang melihat lanskap informasi dengan cara yang sama. Dengan kata lain, desain database adalah alat komunikasi. Berikut merupakan komparasi dari RDBMS dan NoSQL **Table 2.1** (Kunda dan Phiri, 2017).

Tabel 2.1: *Komparasi Database RDBMS dan NoSQL*

No	Kriteria	RDBMS	NoSQL
1	Variety	Terdapat dalam jenis open source dan close platform	Kebanyakan dalam bentuk open source
2	Scalability	Meningkatkan performa dengan meningkatkan hardware dalam server	Meningkatkan dengan cara horizontal atau menambah server
3	Cost	Lebih mahal mengingat harusnya penambahan hardware	Lebih murah mengingat murahnya upgrade secara horizontal
4	Volume of Data	Menangani data yang terbatas	Menangani data yang lebih besar
5	Performance	Lebih lambat karena perlunya memproses informasi	Memiliki query performance lebih baik
6	Complexity	Lebih kompleks karena memerlukan normalisasi hubungan antara tabel	lebih flexible dapat menyimpan struktur yang berbeda dalam satu collections
7	Consistency	Dengan skema yang kompleks menjadikannya lebih konsisten	Kurang konsisten dikarenakan skema yang bermacam-macam
8	Security	Memiliki mekanisme keamanan yang baik untuk melindungi data	Keamanan ditangani oleh middleware dan bukan bagian dari database

D. Flask Framework

Framework adalah kerangka kerja untuk mengembangkan aplikasi berbasis web dan desktop. Kerangka kerja di sini sangat membantu pengembang untuk menulis hal-hal yang lebih terstruktur dan rapi.

Framework ini dibuat untuk menyederhanakan kinerja bagi pemrogram. Jadi programmer tidak perlu menulis kode berulang-ulang. Karena hanya perlu mengkompilasi komponen pemrograman itu sendiri (Adani, 2020).

Flask Framework merupakan kerangka kerja yang berbasis web. Flask dibangun diatas bahasa pemrograman python dengan menggunakan dependensi Werkzeug dan Jinja2. Kerangka kerja Flask bersifat mikro karena tidak membutuhkan alat-alat tertentu atau pustaka. Flask mendukung ekstensi yang dapat menambahkan fitur aplikasi seolah-olah mereka diimplementasikan dalam Flask itu sendiri.

1. Flask routing

Routing adalah modul dalam aplikasi yang mengatur pengoperasian aplikasi berbasis web. Route dapat menangani semua perintah yang telah dideklarasikan. Routing bisa dianalogikan sebagai route diagram penjelas tentang cara menavigasi aplikasi yang sedang dibangun.

App Routing berarti memetakan URL ke fungsi tertentu yang akan menangani logika untuk URL tersebut. Berikut adalah contoh routing di framework flask:

```
from flask import Flask

app = Flask(__name__)

# Pass the required route to the decorator.
@app.route("/hello")
def hello():
    return "Hello, Welcome to GeeksForGeeks"

@app.route("/")
```

```
def index():
      return "Homepage of GeeksForGeeks"

if __name__ == "__main__":
  app.run(debug=True)
```

Seperti potongan code di atas pemetaan URL ke fungsi menggunakan decorator sebelum fungsi ditulis. Dan contoh diatas merupakan contoh yang sederhana pada routing. Berikut adalah beberapa hal yang bisa dilakukan flask routing:

a. Route Methods

Pada dasarnya saat melakukan routing, secara default metode yang dipakai adalah GET. Untuk membuat agar URL dapat menggunakan metode lain seperti POST diharuskan mendefinisikan metodenya pada argumen route.

```
@app.route('/home', methods=['POST', 'GET'])
def home():
    return '<h1>Anda berada di halaman beranda!</h1>'
```

b. Route Variables

Meneruskan variables merupakan sebuah fitur dasar yang memungkinkan pengguna mengirimkan informasi melalui URL. Untuk mengimplementasikan fitur ini route harus mendefinisikan variabel tersebut dalam URL dan menjadikannya argumen pada fungsinya (Rasouli, 2020).

```
@app.route('/home/<firstname>', methods=['POST', 'GET'])
def home(firstname):
    return '<h1>Halo {}, Anda berada di halaman beranda!</ h1>'.format(nama depan)
```

2. Dependencies

Dependencies merupakan ketergantungan suatu sistem kepada sistem lainnya. Dalam halnya framework, ketergantungannya adalah terhadap sebuah package atau extension. Package atau extension berisi sebuah resource suatu logic

yang bisa diimplementasikan dengan mudah dengan hanya memanggil package tersebut yang biasanya berbentuk object class.

a. Flask-REStful

Flask-REStful merupakan extension untuk framework Flask untuk membantu membangun REST API dengan cepat. Flask-REStful mengedepankan peraktek terbaik dengan pengaturan yang diminimalkan (Kevin Burke, 2020).

Pengimplementasian Flask-REStful dimulai dengan penginstalan dengan cara

```
pip install flask-restful
```

dilanjutkan dengan menggunakan import Resource dan Api dari Flask-REStful, buat variabel yang akan menyimpan sebuah class Api dengan argumen Flask, kemudian buat class child dari sebuah class Resource, pada setiap class sudah dapat menerapkan fungsi dari parent Resource seperti get, post, put, dll, kemudian adalah tambahkan mapping class tersebut kepada string endpoint URL.

```
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

Listing II.1: Python example

b. Flask-Mongoengine

Flask-Mongoengine merupakan extension Flask yang menyediakan integrasi Flask ke MongoEngine. MongoEngine sendiri merupakan library python yang berperan sebagai Object Document Mapper dengan MongoDB (Streetlife, 2020).

Pengimplementasian Flask-Mongoengine dimulai dengan penginstalan dengan cara

```
pip install flask-mongoengine
```

dilanjutkan dengan import Mongoengine dari Flask-Mongoengine, lakukan inisiasi untuk app Flask, gunakan fungsi ".config" untuk melakukan konfigurasi database yang akan di pakai, lalu inisiasi MongoEngine dan gunakan fungsi "init_app" dengan memasukkan parameter app Flask.

```
import flask
from flask_mongoengine import MongoEngine


db = MongoEngine()
app = flask.Flask("example_app")
app.config["MONGODB_SETTINGS"] = [
    {
        "db": "project1",
        "host": "localhost",
        "port": 27017,
        "alias": "default",
    }
]
db.init_app(app)
```

Penggunaan MongoEngine diawali dengan mendefinisikan dokumen yang akan di simpan dalam bentuk class yang memiliki parent Document. Mendefinisikan dokumen juga mengharuskan kita mendefinisikan field yang ada didalamnya dengan menetapkannya sebagai property class dan menetapkan jenis dari setiap property class tersebut (Ross Lawley, 2020).

```
from mongoengine import *
import datetime

class Page(Document):
    title = StringField(max_length=200, required=True)
    date_modified = DateTimeField(default=datetime.datetime.utcnow)
```

Setelah Pendefinisian dokumen, memanipulasi data dapat dilakukan dengan memanggil fungsi pada class tersebut. Berikut merupakan contoh dari manipulasi data pada Mongoengine.

```
from mongoengine import *

#Saving Documents
page = Page(title="Test Page")
page.save()
id = page.id

#Update Documents with id
page2 = Page.objects.get(id=id)
page2.update({title: "Test 123"})

#Delete Documents with id
page3 = Page.objects.get(id=id)
page3.delete()
```

E. Apache

Apache adalah perangkat lunak server yang dapat digunakan untuk membuat koneksi antara browser web dan server menggunakan protokol HyperText Transfer, atau HTTP. Apache memiliki fitur yang cukup untuk melakukan konfigurasi dasar seperti pesan kesalahan dan otentikasi berbasis database. Saat digunakan, itu akan membuat kumpulan proses, atau daemon, untuk menangani permintaan. Arsitektur Apache didasarkan pada model client-server berbasis thread yang menggunakan modul; modul ini termasuk keamanan, penulisan ulang URL dan otentikasi kata sandi antara lain (Foundation, 2022).

1. Konfigurasi httpd.conf

Konfigurasi Apache pada server adalah dengan mendaftarkan Aplikasi pada file httpd.conf. Berikut merupakan contoh pendaftaran pada httpd.conf

```
WSGIDaemonProcess /fishapi python-path=/opt/rh/rh-python38/root/lib/pyt$
WSGIProcessGroup /fishapi
WSGIApplicationGroup %{GLOBAL}
WSGIScriptAlias /fishapi /var/www/html/fishapi/index.wsgi
WSGIScriptReloading on
<Directory "/var/www/html/fishapi/fishapi">
  AllowOverride All
  Options +ExecCGI
  AddHandler cgi-script .cgi .pl .py
```

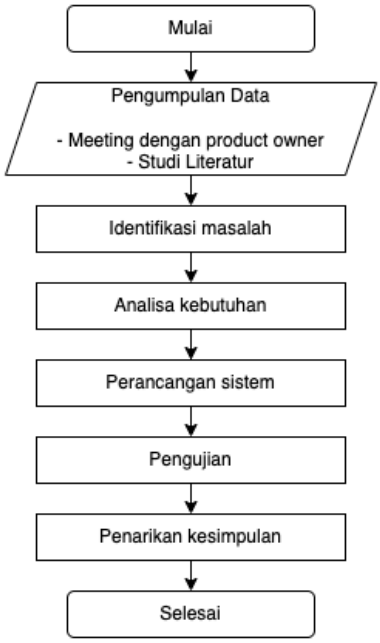
```
  Order allow,deny
  allow from all
</Directory>

Alias /fishapi/static /var/www/html/fishapi/fishapi/static
<Directory /var/www/html/fishapi/fishapi/static>
  AllowOverride All
  Order allow,deny
  allow from all
</Directory>
```

BAB III

Metodologi Penelitian

Penelitian yang dilakukan oleh penulis akan menghasilkan produk tertentu dan akan dilakukan pengujian keefektifannya (Purnama, 2013). Menurut Suhadi Ibnu (Purnama, 2013), penelitian pengembangan bertujuan untuk menghasilkan suatu produk baik itu software ataupun hardware melalui prosedur yangumumnya dimulai dengan menganalisis kebutuhan, kemudian lanjut ke proses pengembangan, dan diakhiri dengan evaluasi.

Berdasarkan pengertian tersebut, penelitian yang akan dilaksanakan oleh penulis masuk ke dalam jenis Penelitian dan Pengembangan/Research and Development. Tahapan penelitian yang akan dilaksanakan penulis dalam perancangan aplikasi dapat dilihat pada **Gambar 3.1**.



Gambar 3.1: Tahapan Peneltian

A. Pengumpulan Data

Data diambil dari hasil wawancara dengan owner J Farm Technology (JFT) Muhammad Eka Suryana, M.Kom., yang sekaligus merupakan klien dari penelitian ini. Selain itu, dilakukan studi literatur dengan membaca jurnal-jurnal yang terkait dengan topik penelitian. Untuk transkrip wawancara dapat dibaca pada Lampiran 1

B. Analisa Kebutuhan

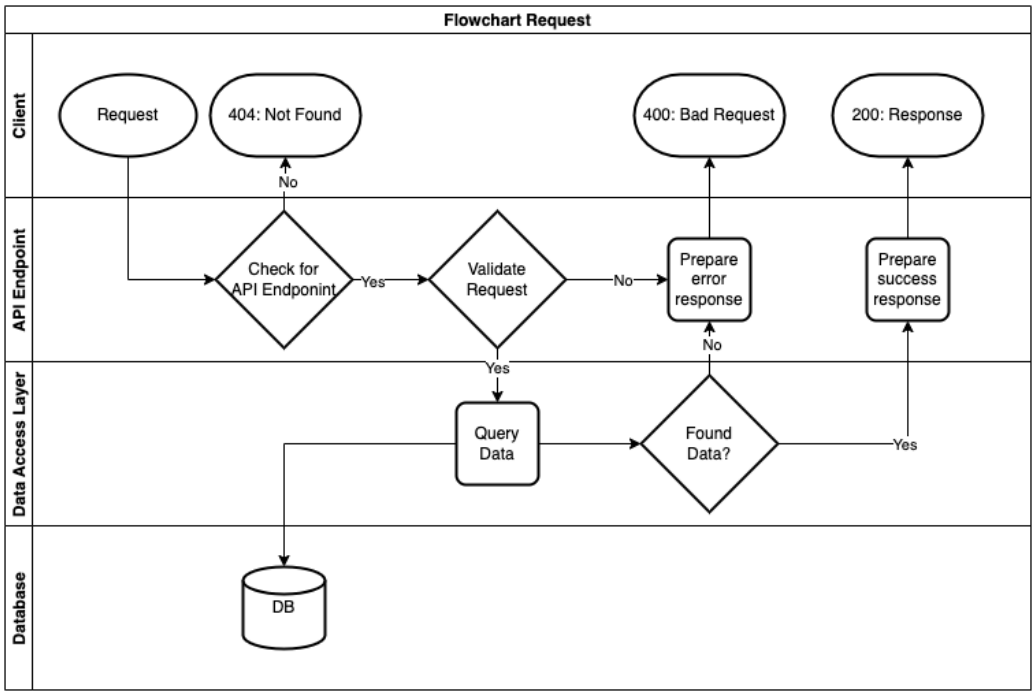
Berdasarkan uraian pada lampiran 1., prioritas fitur pada webservice perikanan modern terfokus pada pencatatan aktifitas pembudidaya ikan seperti pemberian pakan, pencatatan kematian ikan, dll.

Perangkat keras dan perangkat lunak yang penulis butuhkan adalah sebagai berikut:

1. Perangkat keras, terdiri dari:
    a. Laptop dengan spesifikasi Processor Apple M1 & Ram 8 GB
2. Perangkat lunak, terdiri atas:
    a. MacOs ventura 13.0
    b. Visual Studio Code sebagai IDE dan code editor
    c. Python sebagai bahasa pemrograman penyusun aplikasi web service
    d. Flask sebagai framework python
    e. MongoDB sebagai basis data

C. Analisa Sistem

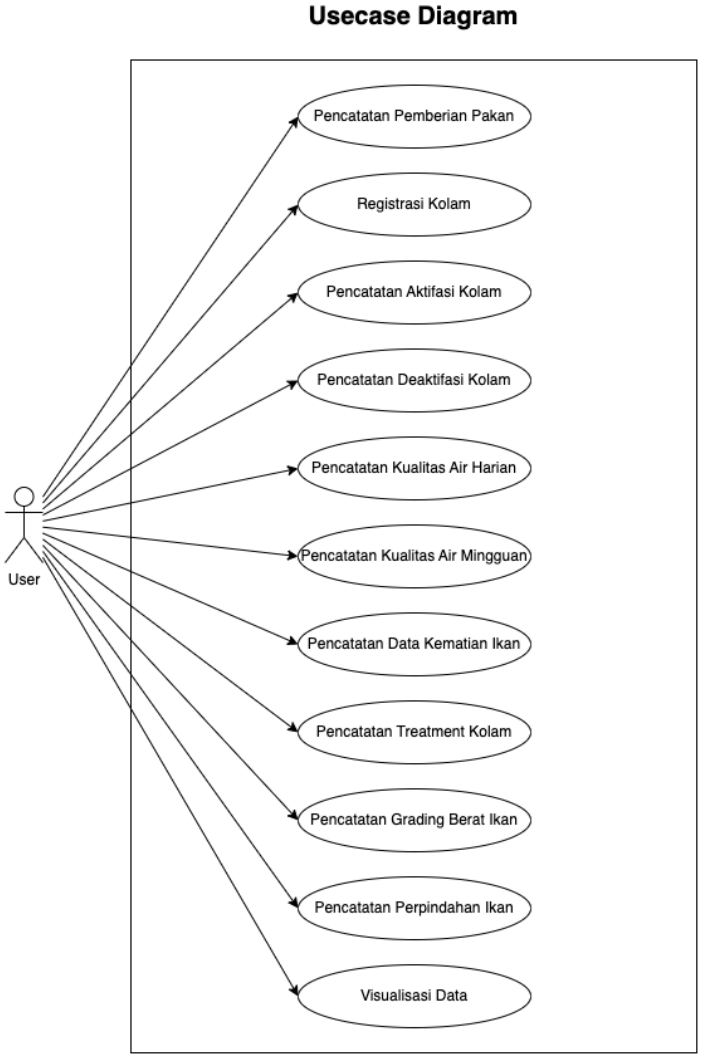
Sistem yang akan dihasilkan oleh penelitian ini berupa sebuah web service, lebih tepatnya adalah Private API. Sistem akan menggunakan arsitektur REST yang mana sering juga disebut RESTful. Sistem akan memberikan respons dari setiap request yang dilakukan oleh client. Request yang dilakukan oleh client memakai protokol http dan response yang diberikan oleh sistem berupa JSON Object. Yang nantinya sistem akan dijalankan di online server agar bisa di akses oleh client menggunakan akses internet.



Gambar 3.2: *Flowchart* request

Dimulai dari request yang dilakukan oleh client, ada beberapa proses yang akan dilakukan oleh sistem. Proses yang dilakukan oleh sistem diantaranya adalah pengecekan API EndPoint, validasi request, pengambilan dan penulisan data di database, dan persiapan response. Pengecekan API EndPoint dilakukan oleh sistem

untuk mengetahui request apa yang diminta oleh client dari URL yang diakses. validasi request dilakukan untuk memastikan data yang dikirim oleh user telah sesuai dengan protokol yang telah ditentukan. Setelahnya proses yang dilakukan adalah memanipulasi atau membaca data di database. dan diakhiri dengan persiapan protokol response sebelum dikirim ke client.



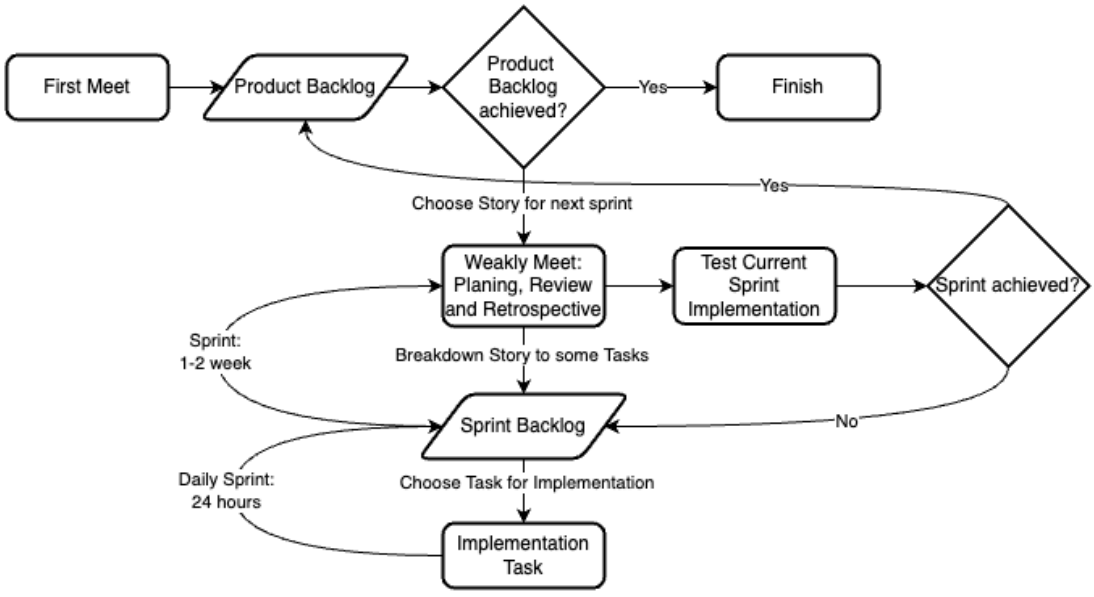
Gambar 3.3: *Usecase* diagram

Gambar 3.3 merupakan usecase diagram user yang dibuat berdasarkan story yang telah didiskusikan bersama *scrum master*. usecase tersebut menggambarkan

kegiatan apa saja yang bisa dilakukan oleh user dalam menggunakan aplikasi. Pada penerapan pada *backend usecase* nantinya akan di pecah lebih mendetail saat mendeskripsikan *task* dari *story*.

D. Perancangan Sistem

Untuk keberlangsungan penelitian, penulis menggunakan metodologi pengembangan sistem scrum Adapun tahapan-tahapan bagian dari scrum yang dilakukan dalam proses penelitian skripsi yang berjudul "Perancangan Arsitektur Backend Server Teknologi Perikanan Modren Berikut Spesifikasi Web Service yang Mampu Berhubungan dengan Multi Platform dengan Metode Scrum". Flowchart metodologi Scrum yang dapat dilihat pada **Gambar 3.4**.



Gambar 3.4: Flowchart Scrum

Tahapan pertama penelitian adalah melakukan meeting dengan product owner dan scrum master. Meeting tersebut bertujuan untuk mendefinisikan product backlog yang akan menjadi acuan sprint kedepannya. Berikutnya merupakan weekly

meeting pertama, agenda pada meeting ini adalah sprint planning. Sprint planning dilakukan dengan cara memilih story dari product backlog yang akan dilakukan breakdown sehingga menghasilkan sprint backlog. Sprint backlog merupakan kumpulan dari task-task kecil yang akan diimplementasi oleh developer setiap harinya. Weekly meeting berikutnya memiliki agenda tambahan dibanding dengan weekly meeting pertama yaitu, adanya sprint review dan sprint retrospective sebelum dilakukannya sprint planning untuk minggu berikutnya. Sprint review dan retrospective merupakan sebuah kajian dari task yang sudah dikerjakan oleh developer dan sprint yang dilakukan bersama oleh scrum master. Apabila tujuan dari sprint itu tercapai, maka scrum master akan menentukan story dari product backlog untuk sprint selanjutnya. Setelah semua product backlog tercapai maka sistem bisa dikatakan telah selesai.

Metode scrum terdiri dari beberapa komponen diantaranya adalah product backlog, sprint backlog, sprint, daily scrum, dan pengujian sistem. Berikut adalah penjelasan dari komponen-komponen tersebut:

1. Produk Backlog

Product backlog dibuat berdasarkan hasil diskusi dengan product owner sekaligus scrum master. Transkrip percakapan diskusi terdapat pada **Lampiran**. Setiap story yang ada di product backlog akan diselesaikan bertahap tiap minggunya. Berikut adalah tabel product backlog.

Tabel 3.1: *Product Backlog*

No	User Story	Sprint	Priority
1	Create, Read, Updte, dan Delete untuk Pencatatan Pemberian pakan	1,2	High
2	Create, Read, Updte, dan Delete untuk Registrasi kolam	3	High
3	Melakukan Aktifasi kolam	4	High
4	Create, Read, Updte, dan Delete untuk Pencatatan data kematian ikan	5,6	Medium
5	Create, Read, Updte, dan Delete untuk Pencatatan Sortir ikan	7	Medium
6	Create, Read, Updte, dan Delete untuk Pencatatan Grading berat ikan	8	Medium
7	Create, Read, Updte, dan Delete untuk Pencatatan kualitas air harian	9	Low
8	Create, Read, Updte, dan Delete untuk Pencatatan kualitas air mingguan	10	Low
9	Create, Read, Updte, dan Delete untuk Pencatatan treatment kolam	11	Low

Dari **Tabel 3.1** di atas, Produk Backlog terdiri dari empat komponen yaitu, User Story, Sprint, dan Priority. Story merupakan sebuah pekerjaan yang menggambarkan suatu fitur atau suatu module, yang mana nantinya akan diuraikan kedalam sprint dalam bentuk task. Komponen sprint merupakan perkiraan pada sprint keberapakah story akan dikerjakan. Priority merupakan sebuah keterangan seberapa prioritasnya user story.

2. Sprint Backlog

Sprint backlog merupakan sebuah task atau pekerjaan yang cenderung lebih kecil atau ringan. Sprint backlog merupakan uraian dari satu story pada product backlog yang menghasilkan beberapa task ringan. Sprint backlog diuraikan oleh scrum master. kemudian, developer bisa memilih task mana yang akan dikerjakan dahulu. Berikut merupakan tabel sprint backlog pertama.

1. Sprint-1

   Sprint-1 dilaksanakan selama delapan minggu yaitu pada tanggal 5 April sampai 31 Mei 2021. delapan minggu tersebut dikarenakan terdapatnya libur lebaran idul fitri selama 2 minggu yaitu pada tanggal 25 April - 9 May. Lalu enam minggu sisanya merupakan waktu yang dibutuhkan penulis untuk menulis penelitian, pembelajaran Scrum, pembelajaran framework flask, dan pembelajaran deployment pada server. Sprint ini merupakan uraian dari story "Pencatatan pemberian pakan" pada product backlog.

Tabel 3.2: *Sprint-1 backlog*

No	***Story***	***Task***	***Status***
1	Create, Read, Updte, dan Delete untuk Pencatatan Pemberian pakan	Merancang Database	Completed
2		Merancang struktur flask framework	Completed
3		Merancang class diagram	Completed
4		Implementasi model Pond, Feed_type, Feed_history	Tested
5		Implementasi controller EndpointApi	Tested
6		Deployment Sistem	Tested
7		Konfigurasi sistem di server	Tested
8		Merancang Dokumentasi API	Tested

3. Daily Scrum

Daily scrum tidak dilakukan secara meeting dikarenakan terbatasnya waktu scrum master dan developer. Maka dari itu, daily scrum digantikan dengan mencatat hambatan dan hal penting yang terjadi pada saat menjalankan sprint setiap harinya pada note github projects.

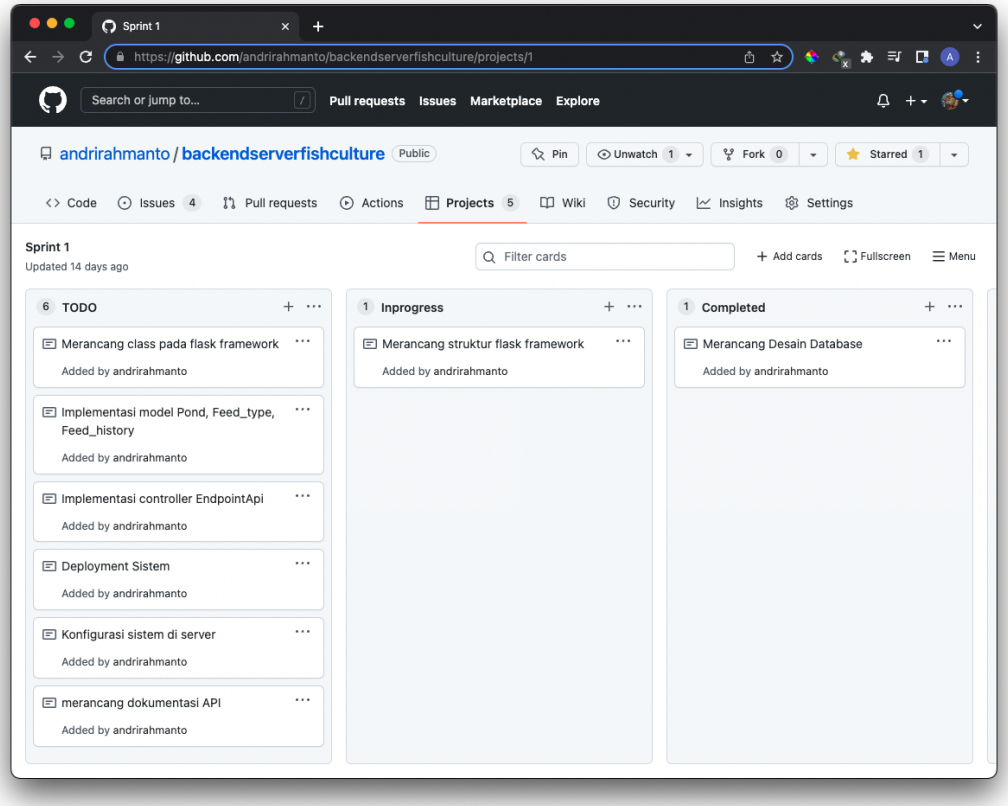
4. Sprint Review dan Sprint Retrospective

Sprint review dan retrospective dilakukan pada awal pekan tepatnya pada hari selasa. Sprint review dan retrospective dilakukan dengan cara voice call melalui platform telegram dan discord. Pembahasan dari meeting ini dimulai dari evaluasi perkembangan sistem, hambatan yang terjadi, dan penetapan sprint kedepannya.

5. Sprint 1 Report

Pada sprint pertama ini story yang di pilih untuk di uraikan pada sprint kali ini adalah pencatatan pemberian pakan. Rentang waktu sprint ini sedikit lebih lama dari sprint lainnya sekitar 2-3 minggu, yang mana sprint lainnya hanya 1-2 minggu. Perbedaan tersebut terjadi karena penulis melakukan training terhadap bahasa dan framework yang akan digunakan dalam pengembangan sistem, sekaligus memberi waktu penulis untuk melakukan pendaftaran dan penulisan pada penelitian.

Tujuan dari sprint-1 adalah membuat CRUD berbentuk API untuk sebuah fitur pencatatan pemberian pakan. Dimulai dari mendesain database, desain struktur sistem, desain class pada sistem, membuat program dari desain yang sudah ada, dan yang terakhir adalah melakukan deployment sistem ke server. Penggunaan platform Github Projects untuk mempermudah manajemen task dan progress seperti pada **Gambar 3.5**.

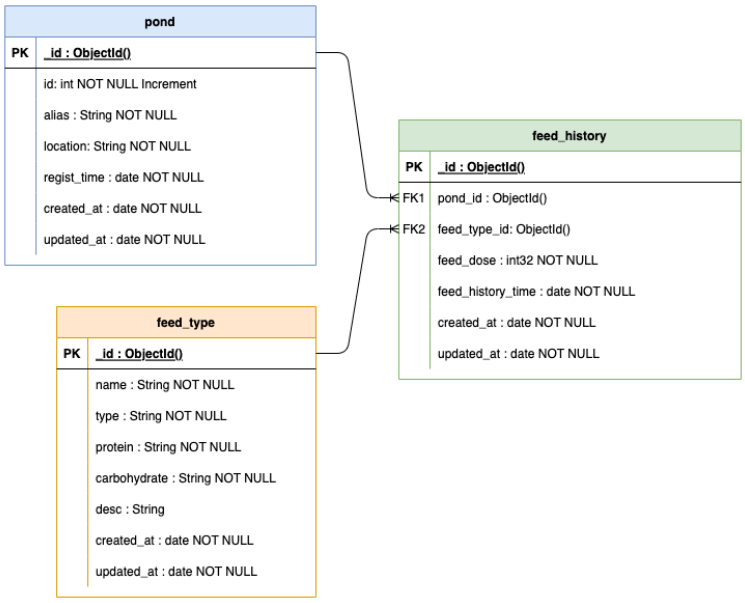


Gambar 3.5: Github Projects Sprint-1

Dari **Gambar 3.5** diatas, terdapat 4 kolom yang menggambarkan fase dari task tersebut. 4 fase itu adalah to do, in progress, completed, dan tested. To do menandakan fase dimana task baru hanya didaftarkan pada list. In progress menandakan fase dimana task sedang dalam proses pengerjaan oleh developer. Completed merupakan fase dimana task sudah selesai dikerjakan. Dan tested merupakan fase dimana task tertentu telah di test oleh developer dan scrum master.

a). Merancang Database

Desain Database atau biasa disebut Entity Relationship Diagram (ERD) pada sprint-1 menggambarkan struktur data dan relasi antar data yang akan disimpan pada database.



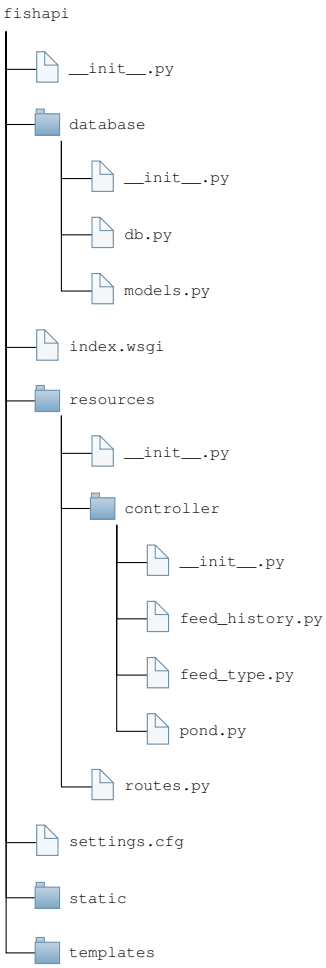
Gambar 3.6: ERD Database Sprint-1

Terdapat 3 tabel yang dibutuhkan untuk menunjang fitur pemberian pakan. Tabel pond tabel pond berfungsi untuk menyimpan data terkait kolam, dimana ada beberapa field yaitu nama dan lokasi kolam. Tabel *feed_type* berfungsi menyimpan data terkait tipe pakan, dimana field yang disimpan adalah nama pakan, tipe pakan, kandungan protein pakan dalam bentuk persentase, kandungan karbohidrat pakan dalam bentuk persentase, dan keterangan tambahan. Tabel *feed_history* berfungsi untuk menyimpan data terkait pemberian pakan, dimana field yang disimpan adalah id kolam, id tipe pakan, dosis pakan, dan waktu pemberian pakan.

b). Merancang Struktur Flask Framework

Rancangan struktur flask framework menggambarkan direktori-direktori pada sistem yang akan di buat. terdapat 4 direktori penting yaitu database, resource, static, template. Direktori database berisi file yang berkaitan dengan database mongodb yaitu, koneksi database dan model data pada database. Direktori

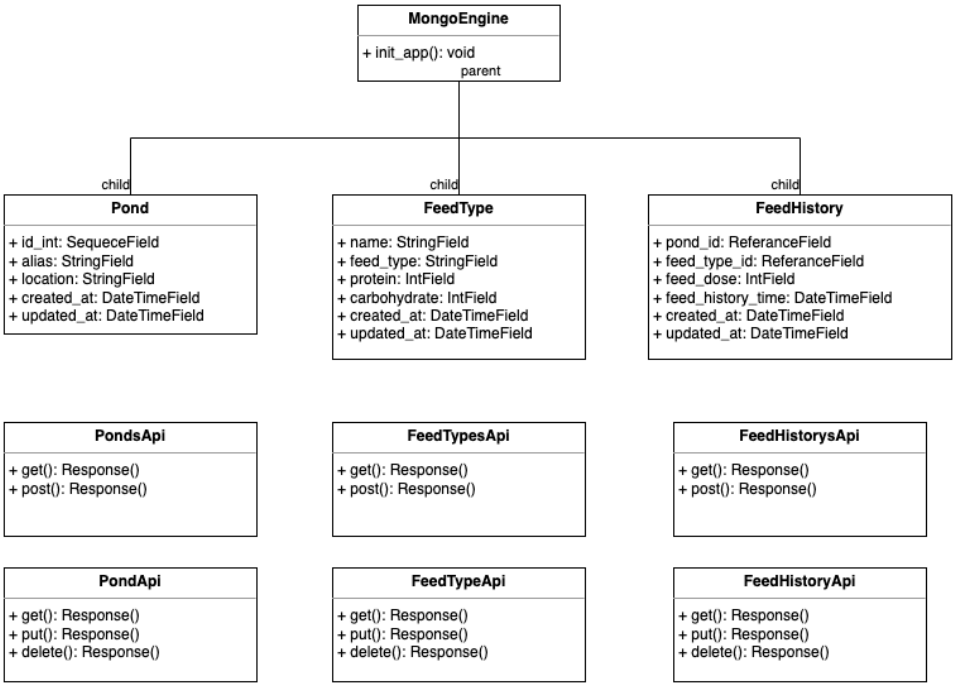
resource deskripsi routes dan controller untuk setiap logic API endpoint. Direktori static berisi hal-hal yang dibutuhkan untuk styling sebuah halaman html, yang biasanya berisi css file dan javascript file. Direktori template merupakan tempat penulis menyimpan html file. Pembagian direktori tersebut bertujuan untuk mempermudah pengolahan dalam pengembangan sistem kedepannya.



c). Merancang Class Diagram

Class Diagram menggambarkan kelas-kelas yang akan dipakai oleh sistem. Umumnya terdapat 3 kelas pada setiap module yaitu class model, controller, dan view. Untuk sprint-1 setiap modul hanya terdapat 2 class yaitu model, dan

controller.



Gambar 3.7: Class Diagram Sprint-1

Pada **Gambar 3.7** setiap module terdapat 3 class. Semisal module Pond terdapat 1 class untuk model dan 2 class controller API. class model merupakan class yang mewarisi sifat MongoEngine.Document yang merupakan sebuah package pada python untuk mempermudah manipulasi database. Sedangkan class controller API merupakan class yang dibuat untuk mendeskripsikan logic yang akan dijalankan setiap client mengakses API Endpoint.

d). Implementasi model Pond, Feed_type, Feed_history

Implementasi model dilakukan menggunakan bantuan package Mongoengine yang mana mengharuskan menjadikan Mongoengine.Document sebagai parent dari class tersebut. Pada setiap atribut class harus didefinisikan terhadap jenis field yang telah disediakan oleh Mongoengine.

1). Model Pond

```
class Pond(db.Document):
    id_int = db.SequenceField(required=True)
    alias = db.StringField(required=True)
    location = db.StringField(required=True)
    created_at = db.DateTimeField(default=datetime.datetime.now)
    updated_at = db.DateTimeField(default=datetime.datetime.now)

```

2). Model Feed_type

```
class FeedType(db.Document):
    name = db.StringField(required=True)
    feed_type = db.StringField(required=True)
    protein = db.IntField(required=True)
    carbohydrate = db.IntField(required=True)
    desc = db.StringField()
    created_at = db.DateTimeField(default=datetime.datetime.now)
    updated_at = db.DateTimeField(default=datetime.datetime.now)

```

3). Model Feed_history

```
class FeedHistory(db.Document):
    pond_id = db.ReferenceField(Pond, required=True)
    feed_type_id = db.ReferenceField(FeedType, required=True)
    feed_dose = db.IntField(required=True)
    feed_history_time = db.DateTimeField(default=datetime.datetime.now)
    created_at = db.DateTimeField(default=datetime.datetime.now)
    updated_at = db.DateTimeField(default=datetime.datetime.now)

```

e). Implementasi Controller EndpointAPI

1). Add Routes

```
def initialize_routes(api):
    # pond
    api.add_resource(PondsApi, '/api/ponds')
    api.add_resource(PondApi, '/api/ponds/<id>')

```

2). PondsApi Class PondsApi

```
class PondsApi(Resource):
    def get(self):
        objects = Pond.objects()
        response = []
        for pond in objects:
            pond = pond.to_mongo()
```

```
            response.append(pond)
        response_dump = json.dumps(response, default=str)
        return Response(response_dump, mimetype="application/json", status=200)

    def post(self):
        body = {
            "alias": request.form.get("alias", None),
            "location": request.form.get("location", None),
        }
        pond = Pond(**body).save()
        id = pond.id
        return {'id': str(id)}, 200

```

3). PondApi

```
class PondApi(Resource):
    def put(self, id):
        body = {
            "alias": request.form.get("alias", None),
            "location": request.form.get("location", None),
            "updated_at": datetime.datetime.utcnow()
        }
        Pond.objects.get(id=id).update(**body)
        return '', 200

    def delete(self, id):
        pond = Pond.objects.get(id=id).delete()
        return '', 200

    def get(self, id):
        objects = Pond.objects.get(id=id)
        pond = objects.to_mongo()
        response_dump = json.dumps(pond, default=str)
        return Response(response_dump, mimetype="application/json", status=200)

```

f). Deployment Sistem

Sistem pada akhirnya dijalankan pada server agar dapat diakses client melalui akses internet. Pertama yang dilakukan adalah mengunggah repositori sistem ke repositori github. Repositori github terdapat pada ***https://github.com/andrirahmanto/backendserverfishculture/tree/sprint_01***.

Lalu setelahnya adalah menghubungkan personal computer penulis ke server dengan metode SSH (Secure Shell Connection). Menghubungkan dengan metode ssh dilakukan melalui terminal pada dengan cara

```
ssh <user>@<ip_address_server>
```

dan masukan password ketika diminta.

Selanjutnya hubungkan server dengan repositori github yang sudah dibuat menggunakan metode git remote.

```
mkdir /var/www/html/<name_your_repo>
git init
git remote add origin <link_github>
git pull origin main
```

dengan begitu server telah memiliki repo yang sama dengan yang ada di personal komputer. Langkah selanjutnya diikuti dengan penginstalan package yang dipakai pada sistem.

g). Konfigurasi Sistem pada Server

Konfigurasi bertujuan untuk menjalankan sistem yang sudah di upload ke server. Tahapan pertama konfigurasi adalah memastikan bahwa sistem berbentuk package agar dapat di import. kemudian buat file index.wsgi seperti dibawah ini

```
import logging
import sys

logging.basicConfig(stream=sys.stderr)
sys.path.insert(0, '/var/www/html/fishapi')

from fishapi import create_app
application = create_app()
```

index.wsgi nantinya akan didaftarkan ke file httpd.conf pada server agar sistem dapat dieksekusi oleh server.

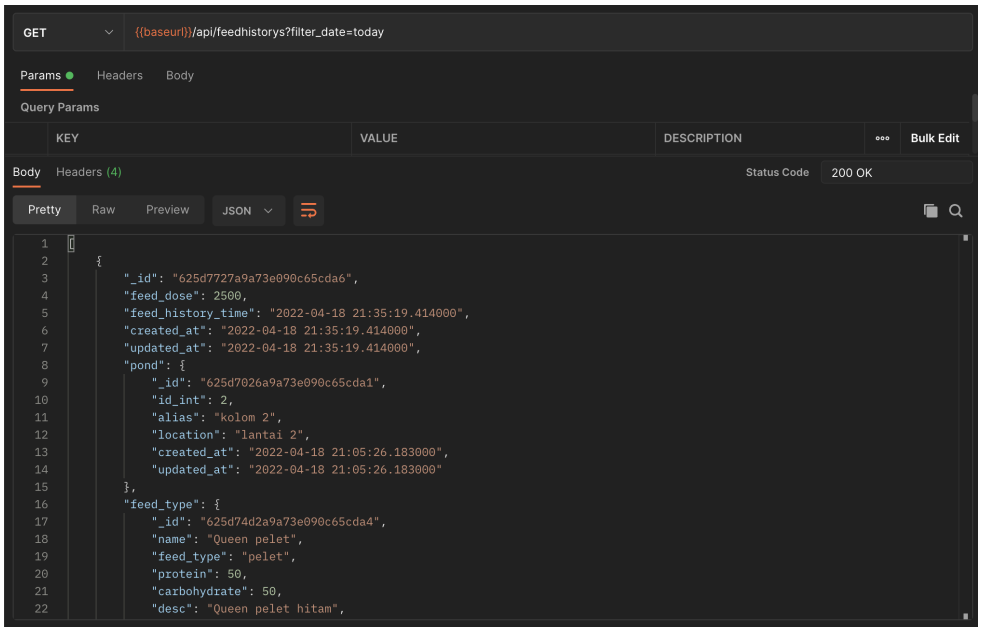
```
WSGIDaemonProcess /fishapi python-path=/opt/rh/rh-python38/root/lib/pyt$
WSGIProcessGroup /fishapi
WSGIApplicationGroup %{GLOBAL}
WSGIScriptAlias /fishapi /var/www/html/fishapi/index.wsgi
WSGIScriptReloading on
<Directory "/var/www/html/fishapi/fishapi">
  AllowOverride All
  Options +ExecCGI
  AddHandler cgi-script .cgi .pl .py
  Order allow,deny
```

```
  allow from all
</Directory>

Alias /fishapi/static /var/www/html/fishapi/fishapi/static
<Directory /var/www/html/fishapi/fishapi/static>
  AllowOverride All
  Order allow,deny
  allow from all
</Directory>
```

Setelahnya lakukan restart pada apache pada server dengan menjalankan:

```
systemctl restart http
```

Setelah konfigurasi selesai, sistem akan di test pada browser dengan melakukan request.



Gambar 3.8: *Request Get* pemberian pakan *Postman*

h). Dokumentasi API

Dokumentasi API dibuat menggunakan postman, yang mana dapat diakses melalui link ***https://documenter.getpostman.com/view/11714934/UVysybzY***. berikut beberapa contoh request dan response yang diberikan server: request cURL:

```
curl --location --request GET 'http://jft.web.id/fishapi/api/ponds'
```

response json:


```
[
  {
    "_id": "625d7026a9a73e090c65cda1",
    "id_int": 2,
    "alias": "alpha",
    "location": "blok 1",
    "shape": "bundar",
    "material": "beton",
    "length": 0,
    "width": 0,
    "diameter": 1.4,
    "height": 1,
    "build_at": "2022-04-18 21:05:26.183000",
    "image_name": "kolam_1655141767.jpeg",
    "area": 1.5399999999999998,
    "image_link": "http://127.0.0.1:5000/api/ponds/image/625d7026a9a73e090c65cda1",
    "volume": 1.5399999999999998
  },
  { "_id": "625d7033a9a73e090c65cda2",.....},
  { "_id": "62a62163e445ffb9c5f746f3",.....},
  { "_id": "62a955888911334402ddb3b3",.....},
  {"_id": "62a9a466299f257e382a8295",.....}
]
```


6. Meeting Sprint 2

Sprint 1 ditutup sebelum sprint 2 dimulai pada weekly meeting hari Selasa. Agenda meeting sprint 2 adalah:

a). Sprint 1 Review

Scrum master memberikan beberapa hal kecil yang harus diperbaiki pada dokumentasi API, tepatnya adalah penambahan keterangan field setiap response.

b). Sprint 2 Planning

Planning untuk sprint 2 adalah menambahkan view visualisasi data "pemberian pakan"

LAMPIRAN A

Transkrip Percakapan

Hari: Selasa

Tanggal: 19 April 2022

PL: Penulis

KL: Klein(Pemilik Farm)

PL: Sistem apa yang akan di buat?

KL: Kita akan membuat *Backend server* berbentuk *REST API* untuk perikanan modren

PL: Apa saja requirement dan fitur yang dibutuhkan oleh sistem ini?

KL: Fitur utama yang harus ada adalah pencatatan pemberian pakan, registrasi kolam, aktivasi-deaktivasi kolam, pencatatan kualitas air harian dan mingguan, pencatatan data kematian harian, pencatatan treatment kolam, *grading* berat ikan kolam, perpindahan ikan antar kolam

PL: Dimulai dari manakah pengerjaan fitur2 tersebut?

KL: Dimulai dari pemberian pakan

DAFTAR PUSTAKA

Adani, M. R. (2020). Pengenalan apa itu framework dan jenisnya untuk web development. https://www.sekawanmedia.co.id/blog/pengertian-framework.

Aep Setiawan, Erlin Arlitasari, M. Z. A. H. (2022). Monitoring pemberian pakan ikan otomatis menggunakan iot di labolatorium perikanan sekolah vokasi ipb. *JINTEKS (Jurnal Informatika Teknologi dan Sains)*.

Ahmad Rifa'i, M. Udin Harun Al Rasyid, A. I. G. (2021). Sistem pemantauan dan kontrol otomatis kualitas air berbasis iot menggunakan platform node-red untuk budidaya udang. *JTT (Jurnal Teknologi Terapan)*.

Alim, H. (2022). Fish movement tracking dengan menggunakan metode gmm dan kalman filter. *Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitar Negeri Jakarta 2022*.

Dhita Widhiastika, Sobakhul Munir Siroj, A. W. U. B. A. P. N. F. R. (2021). Perancangan aplikasi jual beli produk perikanan berbasis mobile android (studi kasus : Fo-klik). *Jurnal Ilmu Perikanan dan Kelautan Indonesia*.

Foundation, A. S. (2022). Apache http server. https://httpd.apache.org/.

Hadi, F. P. (2021). Rancang bangun web service dan website sebagai storage engine dan monitoring data sensing untuk budidaya ikan air tawar. *Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitar Negeri Jakarta 2021*.

Hema Krishnan, T. Santhanakrishnan, M. E. (2016). Mongodb – a comparison with nosql databases). *International Journal of Scientific and Engineering Research*.

infishta (2019). Jenis-jenis budaya ikan air tawar. https://infishta.com/blogs/jenis-jenis-budidaya-ikan-air-tawar.

Kevin Burke, Kyle Conroy, R. H. (2020). Flaskrestful. https://flask-restful.readthedocs.io/en/latest.

Kunda, D. dan Phiri, H. (2017). A comparative study of nosql and relational database.

Nugraha, B. (2022). Ekstrasi latar depan pada citra ikan dengan metode grabcut yang diautomasi menggunakan saliency map. *Program Studi Ilmu Komputer Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitar Negeri Jakarta 2022*.

Pramleonita, M., Yuliani, N., Arizal, R., dan Wardoyo, S. E. (2018). Parameter fisika dan kimia air kolam ikan nila hitam (oreochromis niloticus). *Jurnal Sains Natural*, (1):24–34.

Purnama, S. (2013). Metode penelitian dan pengembangan (pengenalan untuk mengembangkan produk pembelajaran bahasa arab). *LITERASI*.

Rasouli, M. (2020). Flask route a complete overview. https://medium.com/analytics-vidhya/flask-route-a-complete-overview-d66065bfa867.

Ross Lawley, Marr, B. G. (2020). Mongoengine user documentation. https://docs.mongoengine.org.

Streetlife (2020). Flask-mongoengine documentation. http://docs.mongoengine.org/projects/flask-mongoengine/en/latest.

Supriyati, D. M. R. (2018). Model perancangan sistem informasi akuntansi budidaya perikanan berbasis sak emkm dan android. *Program Studi Komputerisasi Akuntansi Fakultas Teknik dan Ilmu Komputer-Universitas Komputer Indonesia*.

Yudhis Thiro Kabul Yunior, K. (2019). Sistem monitoring kualitas air pada budidaya perikanan berbasis iot dan manajemen data. *Creative Information Technology Journal (CITEC JOURNAL)*.